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 3

Complexity Analysis of Adaptivity

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

— B.D. McCullough (2000)

Coming Up Next

1. Analysis of Adaptive Complexity
2. Analysis of Descartes Method
3. Integral Bounds and Framework of Stopping Functions

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Towards Analysis of Adaptive Algorithms

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

Analytic Approach to Root Isolation

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

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

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

STURM > DESCARTES > BOLZANO

Analytic Approach to Root Isolation

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

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

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

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

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

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

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

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

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

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

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

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

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

$$STURM > DESCARTES > BOLZANO$$

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

```
Let Q_in ← {I_0} be a queue
WHILE (Q ≠ ∅)                              ◁ Subdivision Phase
        I ← Q.remove()
        IF (C_0(I) holds), discard I
        ELIF (C_1(I) holds), output I
        ELSE
                IF (f(m(I)) = 0), output [m(I), m(I)]
                Split I into two and insert in Q
PROCESS output list                        ◁ Consruction Phase
```

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

EVAL Algorithm

EVAL

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

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

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

Warm Up Technique: Algebraic Amortization

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

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

Warm Up Technique: Algebraic Amortization

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

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

Warm Up Technique: Algebraic Amortization

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

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

Warm Up Technique: Algebraic Amortization

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

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

Warm Up Technique: Algebraic Amortization

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

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

Warm Up Technique: Algebraic Amortization

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

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

The Davenport–Mahler Bound

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

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

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

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

(iii) *G is acyclic,*

then

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

where

$$\mathrm{discr}(A) := a_n^{2n-2} \prod_{i>j} (\vartheta_i - \vartheta_j)^2 \quad \textit{and} \quad \mathrm{M}(A) := |a_n| \prod_i \max\{1, |\vartheta_i|\}.$$

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Coming Up Next

1 Analysis of Adaptive Complexity

2 Analysis of Descartes Method

3 Integral Bounds and Framework of Stopping Functions

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $\mathit{DescartesTest}(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $\mathit{DescartesTest}(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

Analysis of Descartes Method

Two-circle Theorem

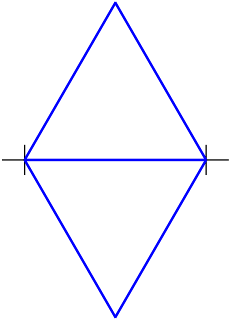
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If DescartesTest$(A(X),[c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α,β of $A(X)$.

Corollary

Can choose α,β to be complex conjugate or adjacent real roots.
Moreover, $|\beta-\alpha| < \sqrt{3}(d-c)$; i.e., $(d-c) > |\beta-\alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

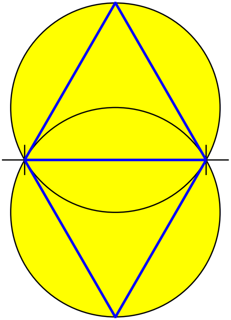
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If DescartesTest $(A(X),[c,d]) \geq 2$, *then the two-circles figure in* $\mathbb{C}$ *around interval* $[c,d]$ *contains two roots* α, β *of* $A(X)$.

Corollary

Can choose α, β *to be complex conjugate or adjacent real roots.*
Moreover, $|\beta - \alpha| < \sqrt{3}(d-c)$; *i.e.,* $(d-c) > |\beta - \alpha| / \sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

[Ostrowski (1950), Krandick/Mehlhorn (2006)]

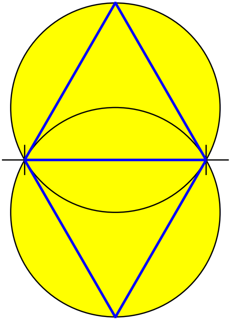
If $DescartesTest(A(X),[c,d]) \geq 2$*, then the two-circles figure in* $\mathbb{C}$ *around interval* $[c,d]$ *contains two roots* α,β *of* $A(X)$*.*

Corollary

Can choose α,β *to be complex conjugate or adjacent real roots.*

Moreover, $|\beta-\alpha| < \sqrt{3}(d-c)$*; i.e.,* $(d-c) > |\beta-\alpha|/\sqrt{3}$*.*

Analysis of Descartes Method



Two-circle Theorem

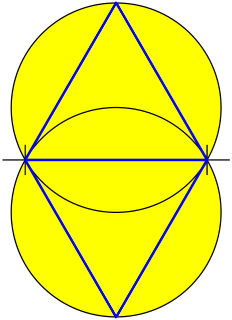
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If DescartesTest $(A(X),[c,d]) \geq 2$, *then the two-circles figure in* $\mathbb{C}$ *around interval* $[c,d]$ *contains two roots* α,β *of* $A(X)$.

Corollary

Can choose α,β *to be complex conjugate or adjacent real roots.*
Moreover, $|\beta-\alpha| < \sqrt{3}(d-c)$; *i.e.,* $(d-c) > |\beta-\alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

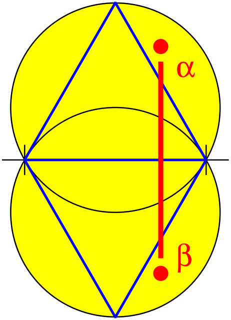
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If DescartesTest$(A(X),[c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α,β of $A(X)$.

Corollary

Can choose α,β to be complex conjugate or adjacent real roots.
Moreover, $|\beta-\alpha|<\sqrt{3}(d-c)$; i.e., $(d-c)>|\beta-\alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

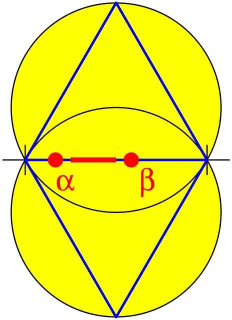
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If DescartesTest$(A(X),[c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α,β of $A(X)$.

Corollary

Can choose α,β to be complex conjugate or adjacent real roots.
Moreover, $|\beta-\alpha| < \sqrt{3}(d-c)$; i.e., $(d-c) > |\beta-\alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

[Ostrowski (1950), Krandick/Mehlhorn (2006)]

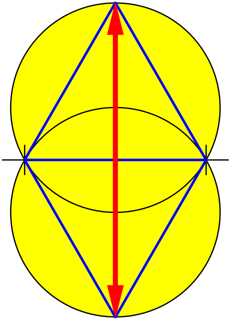
If DescartesTest$(A(X), [c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α, β of $A(X)$.

Corollary

Can choose α, β to be complex conjugate or adjacent real roots.

Moreover, $|\beta - \alpha| < \sqrt{3}(d-c)$; i.e., $(d-c) > |\beta - \alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

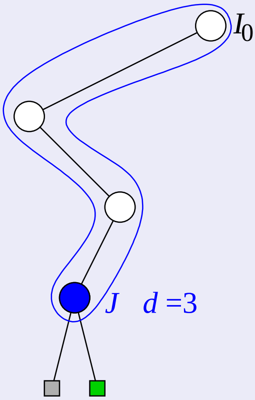
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If DescartesTest $(A(X), [c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α, β of $A(X)$.

Corollary

Can choose α, β to be complex conjugate or adjacent real roots.
Moreover, $|\beta - \alpha| < \sqrt{3}(d-c)$*; i.e.,* $(d-c) > |\beta - \alpha|/\sqrt{3}$.

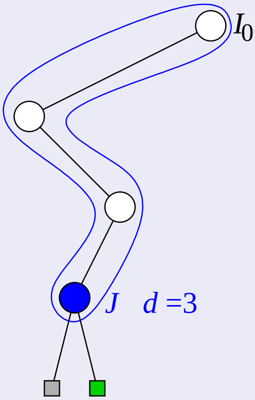
Tree Bound in terms of Roots (1)



A bound on path length

1. Consider any path in the recursion tree from I_0 to a parent J of two leaves.
2. At depth d, interval width is $2^{-d}|I_0|$. Hence depth of J is $d = \log|I_0|/|J|$.
3. The path consists of $d+1$ internal nodes.
4. There is a pair of roots (α_J, β_J) such that $|J| > |\beta_J - \alpha_J|/\sqrt{3}$; hence
$$\boxed{d + 1 < \log|I_0| - \log|\beta_J - \alpha_J| + 2.}$$

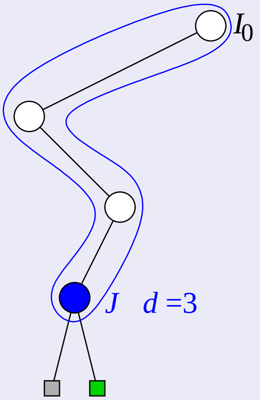
Tree Bound in terms of Roots (1)



A bound on path length

1. Consider any path in the recursion tree from I_0 to a parent J of two leaves.
2. At depth d, interval width is $2^{-d}|I_0|$. Hence depth of J is $d = \log|I_0|/|J|$.
3. The path consists of $d+1$ internal nodes.
4. There is a pair of roots (α_J, β_J) such that $|J| > |\beta_J - \alpha_J|/\sqrt{3}$; hence $\boxed{d+1 < \log|I_0| - \log|\beta_J - \alpha_J| + 2.}$

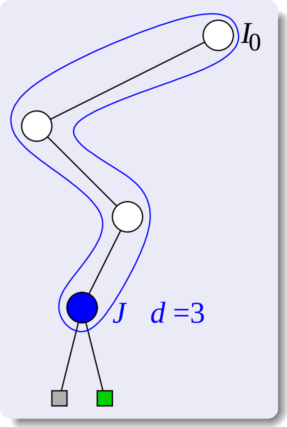
Tree Bound in terms of Roots (1)



A bound on path length

1. Consider any path in the recursion tree from I_0 to a parent J of two leaves.
2. At depth d, interval width is $2^{-d}|I_0|$. Hence depth of J is $d = \log|I_0|/|J|$.
3. The path consists of $d+1$ internal nodes.
4. There is a pair of roots (α_J, β_J) such that $|J| > |\beta_J - \alpha_J|/\sqrt{3}$; hence $\boxed{d+1 < \log|I_0| - \log|\beta_J - \alpha_J| + 2.}$

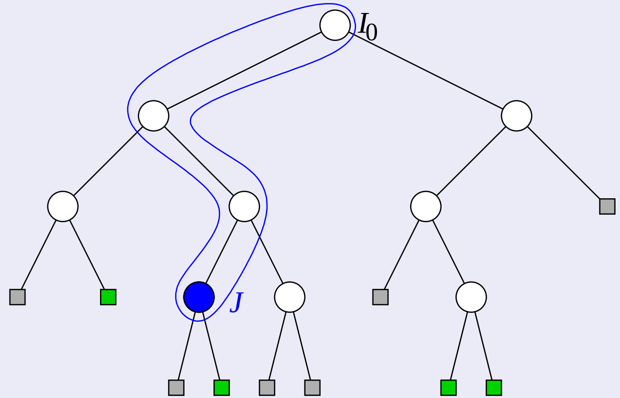
Tree Bound in terms of Roots (1)



A bound on path length

1. Consider any path in the recursion tree from I_0 to a parent J of two leaves.
2. At depth d, interval width is $2^{-d}|I_0|$. Hence depth of J is $d = \log|I_0|/|J|$.
3. The path consists of $d+1$ internal nodes.
4. There is a pair of roots (α_J, β_J) such that $|J| > |\beta_J - \alpha_J|/\sqrt{3}$; hence $\boxed{d+1 < \log|I_0| - \log|\beta_J - \alpha_J| + 2.}$

Tree Bound in terms of Roots (2)

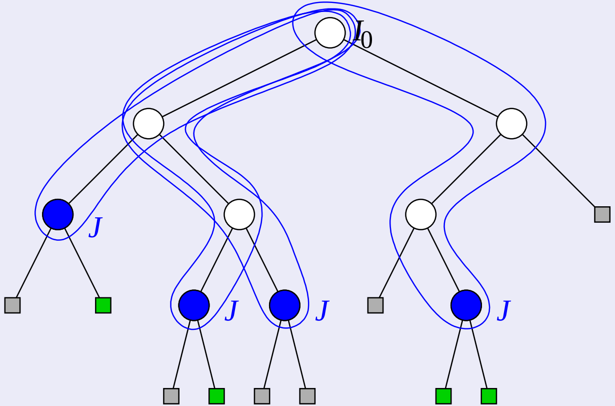


$$\#(\text{internal nodes on path}) < \log|I_0| - \log|\beta_J - \alpha_J| + 2$$

$$\#(\text{internal nodes in tree}) < \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2)$$

$$\#(\text{all nodes in tree}) < 1 + 2 \cdot \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2)$$

Tree Bound in terms of Roots (2)

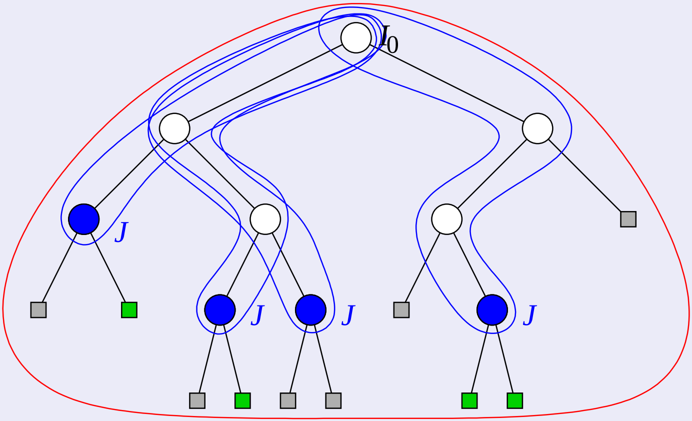


#(internal nodes on path) $<$ $\log|I_0| - \log|\beta_J - \alpha_J| + 2$

#(internal nodes in tree) $<$ $\sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2)$

#(all nodes in tree) $<$ $1 + 2 \cdot \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2)$

Tree Bound in terms of Roots (2)



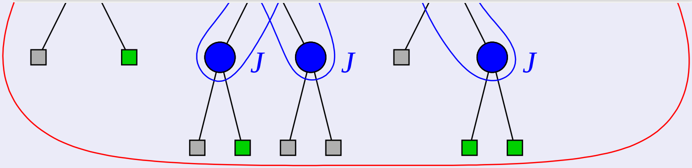
$$\begin{array}{lcr}
\#(\text{internal nodes on path}) & < & \log|I_0| - \log|\beta_J - \alpha_J| + 2 \\
\#(\text{internal nodes in tree}) & < & \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2) \\
\#(\text{all nodes in tree}) & < & 1 + 2 \cdot \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2)
\end{array}$$

Tree Bound in terms of Roots (2)

Proposition

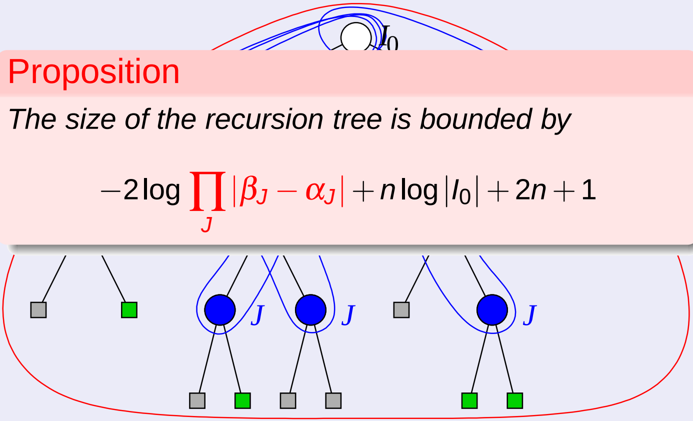
The size of the recursion tree is bounded by

$$-2\log\prod_J|\beta_J-\alpha_J|+n\log|I_0|+2n+1$$



$$\begin{array}{lcr}
\#(\text{internal nodes on path}) & < & \log|I_0|-\log|\beta_J-\alpha_J|+2 \\
\#(\text{internal nodes in tree}) & < & \sum_J(\log|I_0|-\log|\beta_J-\alpha_J|+2) \\
\#(\text{all nodes in tree}) & < & 1+2\cdot\sum_J(\log|I_0|-\log|\beta_J-\alpha_J|+2)
\end{array}$$

Tree Bound in terms of Roots (2)



Proposition

The size of the recursion tree is bounded by

$$-2\log\prod_J|\beta_J-\alpha_J|+n\log|I_0|+2n+1$$

$$
\begin{aligned}
\#(\text{internal nodes on path}) &< \log|I_0|-\log|\beta_J-\alpha_J|+2\\
\#(\text{internal nodes in tree}) &< \textstyle\sum_J(\log|I_0|-\log|\beta_J-\alpha_J|+2)\\
\#(\text{all nodes in tree}) &< 1+2\cdot\textstyle\sum_J(\log|I_0|-\log|\beta_J-\alpha_J|+2)
\end{aligned}
$$

Turning our Product into an Admissible Graph

We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)

Conditions on $G = (V, E)$

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

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

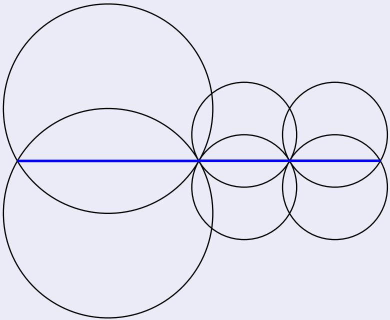
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)



Conditions on $G = (V, E)$

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

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

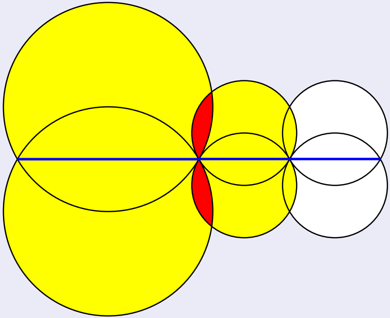
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)



Conditions on $G = (V, E)$

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

(ii) $\beta \in V \implies \text{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

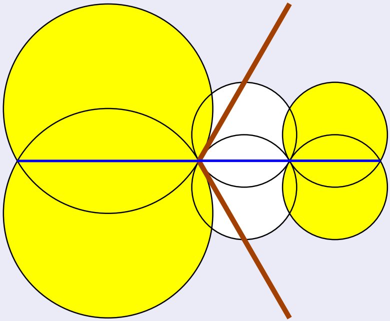
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)



Conditions on $G = (V, E)$

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

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

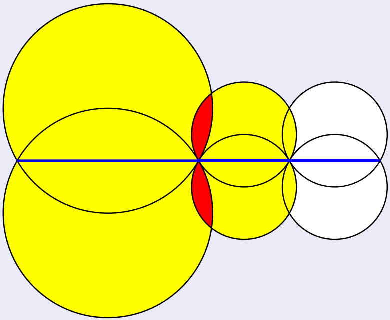
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)



Conditions on $G = (V, E)$

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

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

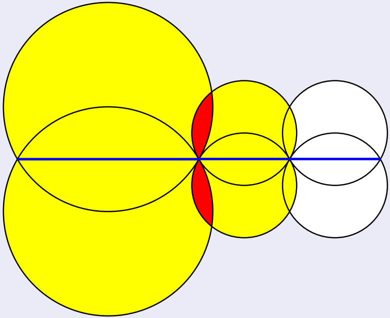
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)

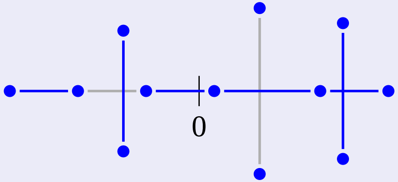


Conditions on $G = (V, E)$

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

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓



Turning our Product into an Admissible Graph

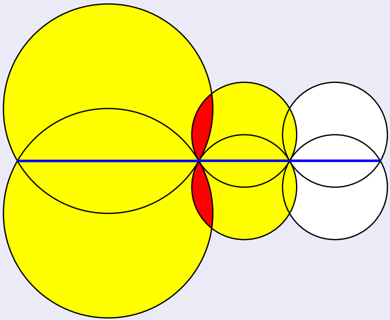
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \quad \text{as} \quad \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate $\leq$ 2

We need two graphs. (Paper: just 1)

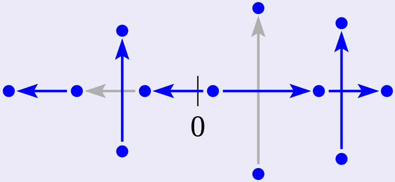


Conditions on $G = (V, E)$

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

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓



Turning our Product into an Admissible Graph

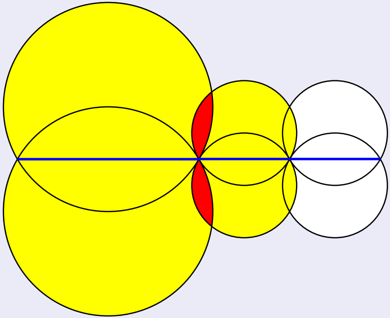
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)

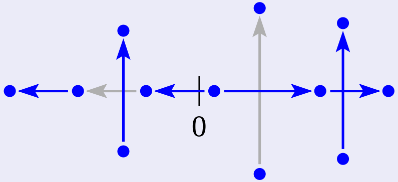


Conditions on $G = (V, E)$

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

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓



Main Result on Descartes Analysis

Theorem (Eigenwillig/Sharma/Y. (2006))

On the Benchmark Problem, we obtain

$$|\mathcal{T}| = O(n(L + \log n)).$$

For $L \geq \log n$, this is optimal.

Argument of [Krandick/Mehlhorn, 2006]: $|\mathcal{T}| = O(n \log n (L + \log n))$.

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - "theory caught up with practice"
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - “theory caught up with practice”
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - “theory caught up with practice”
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - “theory caught up with practice”
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - "theory caught up with practice"
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - "theory caught up with practice"
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - “theory caught up with practice”
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Coming Up Next

1. Analysis of Adaptive Complexity
2. Analysis of Descartes Method
3. Integral Bounds and Framework of Stopping Functions

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$

1. WHILE $(Q \neq \emptyset)$
2. $I \leftarrow Q.remove()$
3. IF $(C(I)$ holds), output I
4. ELSE
5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$

1. WHILE $(Q \neq \emptyset)$
2. $\quad I \leftarrow Q.remove()$
3. $\quad$ IF $(C(I)$ holds), output I
4. $\quad$ ELSE
5. $\quad\quad$ Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$

1. WHILE ($Q \neq \emptyset$)
2. $I \leftarrow Q.remove()$
3. IF ($C(I)$ holds), output I
4. ELSE
5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$

1. WHILE ($Q \neq \emptyset$)
2. $I \leftarrow Q.remove()$
3. IF ($C(I)$ holds), output I
4. ELSE
5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$

1. WHILE ($Q \neq \emptyset$)
2. $\quad I \leftarrow Q.remove()$
3. $\quad$ IF ($C(I)$ holds), output I
4. $\quad$ ELSE
5. $\quad\quad$ Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

If $(\exists b \in I)[w(I) < F(b)]$,

then $C(I)$ holds.

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

If $(\exists b \in I)[w(I) < F(b)]$,

then $C(I)$ holds.

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

$$\text{If } (\exists b \in I)[w(I) < F(b)],$$

$$\text{then } C(I) \text{ holds.}$$

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$

For all interval I:

$$\text{If } (\exists b \in I)[w(I) < F(b)],$$
$$\text{then } C(I) \text{ holds.}$$

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:
$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

$$\text{If } (\exists b \in I)[w(I) < F(b)],$$

$$\text{then } C(I) \text{ holds.}$$

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent (“penultimate leaf”) has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \leq \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \leq \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \leq \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F, H \in \mathbb{Z}[X]$ be relatively prime such that $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ where $\phi, \widetilde{\phi}, \eta, \widetilde{\eta} \in \mathbb{C}[X]$ have degrees $m, \widetilde{m}, n, \widetilde{n}$, respectively. If $\beta_1, \ldots, \beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n}|\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m\left((m+1)\|\phi\|\right)^{\widetilde{n}} M(\widetilde{\eta})^m \left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}} M(H)^{\widetilde{m}}}.$$

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F,H \in \mathbb{Z}[X]$ be relatively prime such that $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ where $\phi,\widetilde{\phi},\eta,\widetilde{\eta} \in \mathbb{C}[X]$ have degrees $m,\widetilde{m},n,\widetilde{n}$, respectively. If $\beta_1,\ldots,\beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n}|\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m\left((m+1)\|\phi\|\right)^{\widetilde{n}} M(\widetilde{\eta})^m\left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}} M(H)^{\widetilde{m}}}.$$

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F, H \in \mathbb{Z}[X]$ be relatively prime such that $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ where $\phi, \widetilde{\phi}, \eta, \widetilde{\eta} \in \mathbb{C}[X]$ have degrees $m, \widetilde{m}, n, \widetilde{n}$, respectively. If $\beta_1, \ldots, \beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n}|\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m\left((m+1)\|\phi\|\right)^{\widetilde{n}} M(\widetilde{\eta})^m \left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}} M(H)^{\widetilde{m}}}.$$

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F, H \in \mathbb{Z}[X]$ be relatively prime such that $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ where $\phi, \widetilde{\phi}, \eta, \widetilde{\eta} \in \mathbb{C}[X]$ have degrees $m, \widetilde{m}, n, \widetilde{n}$, respectively. If $\beta_1, \ldots, \beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n}|\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m\left((m+1)\|\phi\|\right)^{\widetilde{n}} M(\widetilde{\eta})^m \left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}} M(H)^{\widetilde{m}}}.$$

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F, H \in \mathbb{Z}[X]$ be relatively prime such that $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ where $\phi, \widetilde{\phi}, \eta, \widetilde{\eta} \in \mathbb{C}[X]$ have degrees $m, \widetilde{m}, n, \widetilde{n}$, respectively. If $\beta_1, \ldots, \beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n}|\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m\,((m+1)\|\phi\|)^{\widetilde{n}}\,M(\widetilde{\eta})^m\left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}}M(H)^{\widetilde{m}}}.$$

Complex Roots: Lesson from Meshing Curves

How to isolate complex roots?

- Previous subdivision methods:
  - Pan-Weyl Algorithm (Turan Test)
  - Root isolation on boundary of boxes (topological degree)
- Hints from Curve Meshing (Snyder/PV/Cxy) – not good idea

New Result (with Sagraloff)

There is an exact analog CEVAL for complex roots that is simple and easy to implement exactly.
It achieves the same bit complexity bound as in the real case.

Complex Roots: Lesson from Meshing Curves

How to isolate complex roots?

- Previous subdivision methods:
  - Pan-Weyl Algorithm (Turan Test)
  - Root isolation on boundary of boxes (topological degree)
- Hints from Curve Meshing (Snyder/PV/Cxy) – not good idea

New Result (with Sagraloff)

There is an exact analog CEVAL for complex roots that is simple and easy to implement exactly.
It achieves the same bit complexity bound as in the real case.

Complex Roots: Lesson from Meshing Curves

How to isolate complex roots?

- Previous subdivision methods:
  - Pan-Weyl Algorithm (Turan Test)
  - Root isolation on boundary of boxes (topological degree)
- Hints from Curve Meshing (Snyder/PV/Cxy) – not good idea

New Result (with Sagraloff)

There is an exact analog CEVAL for complex roots that is simple and easy to implement exactly.
It achieves the same bit complexity bound as in the real case.

Complex Roots: Lesson from Meshing Curves

How to isolate complex roots?

- Previous subdivision methods:
  - Pan-Weyl Algorithm (Turan Test)
  - Root isolation on boundary of boxes (topological degree)
- Hints from Curve Meshing (Snyder/PV/Cxy) – not good idea

New Result (with Sagraloff)

There is an exact analog CEVAL for complex roots that is simple and easy to implement exactly.
It achieves the same bit complexity bound as in the real case.

Mini Summary

- The Bolzano approach to Root Isolation is an Exact and Analytic approach to root isolation
- It seems to have complexity that matches Sturm and Descartes
- It is much easier to implement than either

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- There are the MANY advantages of numerical/analytic approaches to algebraic and geometric problems
- These methods are practical, adaptive, easy to implement
- The new ingredient we seek is a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open