Reliable and Efficient Geometric Computation

Kurt Mehlhorn
Max-Planck-Institut für Informatik

slides and papers are available at my home page

partially supported by EU-projects

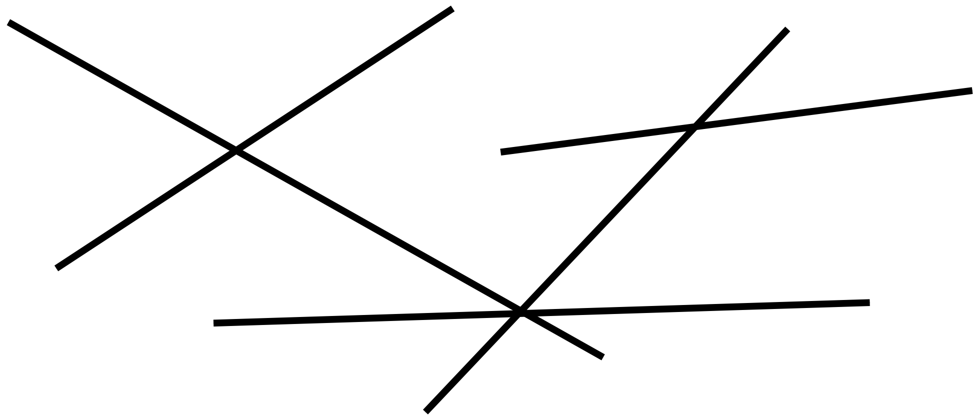
ECG and ACS

Effective Computational Geometry and Algorithms for Complex Shapes)

Overview

- Line Segment Intersection
- Algebraic Curves
- Sweeping Algebraic Curves
- A Crucial Subroutine: Isolating Roots of Real Polynomials
    - Problem Definition
    - A Review of Descartes Algorithm
    - A Randomized Descartes Algorithm
- Sources
    - A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, N. Wolpert: An Exact Descartes Algorithm with Approximate Coefficients, CASC 2005, LNCS 3718, 138–149
    - E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, L. Kettner, K. Mehlhorn, J. Reichel, S. Schmitt, E. Schömer, N. Wolpert :EXACUS—Efficient and Exact Algorithms for Curves and Surfaces, ESA 2005, LNCS 3669, 155-166
    - KM, R. Osbild, M. Sagralof: Reliable and Efficient Computational Geometry via Controlled Perturbation, ICALP 2006

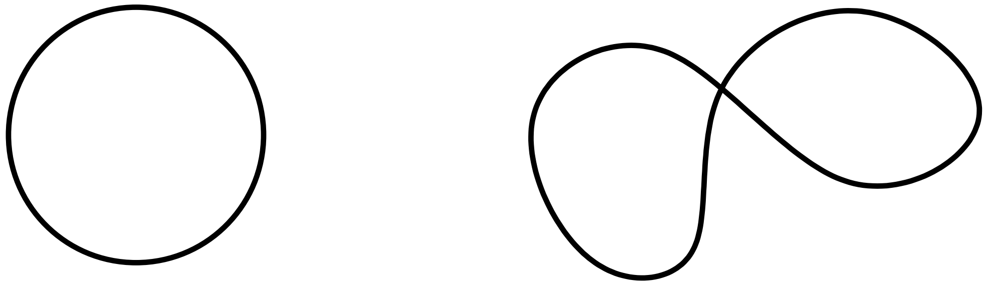
Arrangements of Line Segments



- The sweep-line algorithms solves the problem in time $O(n \log n + s \log n)$
- $n =$ number of segments
- $s =$ number of intersections
- no general position assumption

Arrangements of Algebraic Curves

- algebraic curve = zero set of a polynomial $p(x,y)$



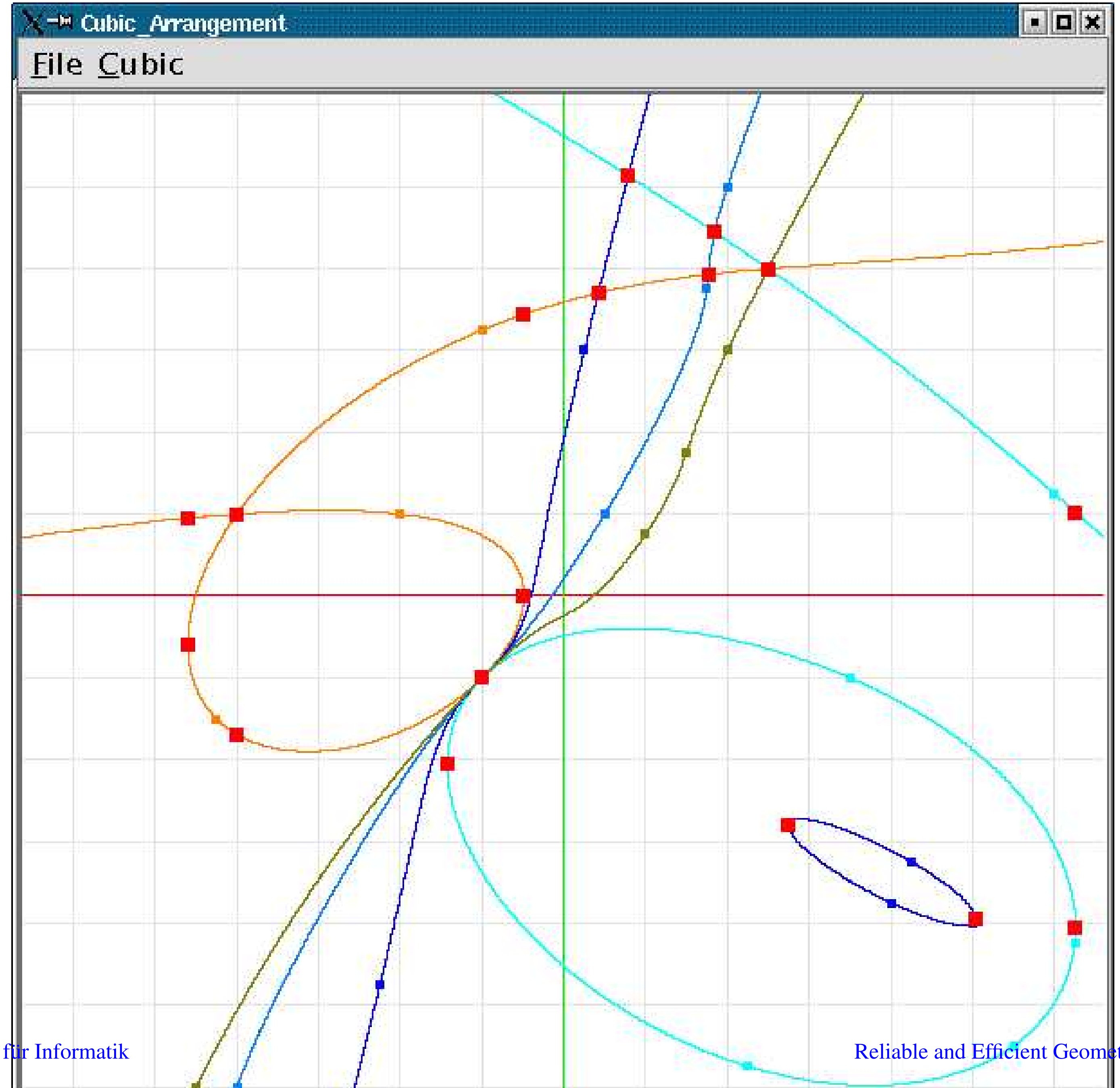
$x^2 + y^2 = 5$ $\qquad\qquad$ $3x^4 + \ldots$

- algorithm readily extends (Chazelle, about 1985)

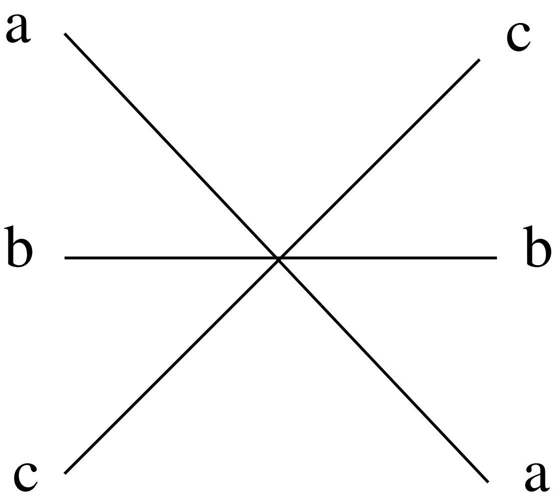
Sweeping Algebraic Curves

1. decompose each curve into x-monotone curve-segments
2. when two curve-segments become adjacent, compute the arrangement defined by the pair of curves
3. everything else stays the same
4. for bounded degree curves, (1) and (2) take constant time by general principles
   - cylindrical algebraic decomposition (Collins)
   - decidability of first order theory of real closed fields (Tarski)
5. $O(n \log n + s \log n)$ with a big big-O

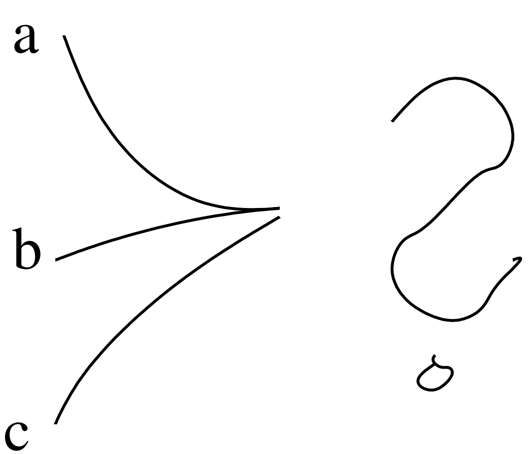
An Arrangement of Cubic Curves



Sweeping a High Degree Vertex



- in the case of *segments*:
  - when we sweep through a vertex, the y-order of the segments is reversed
  - and hence the update of the order is fairly simple
  - linear time in degree of the vertex



- in the case of *curves*:
  - when we sweep through a vertex, the y-order of the curves changes according to an arbitrary permutation
  - or maybe not so arbitrary ?

Two Curves

- intersections of $p(x,y) = 0$ and $q(x,y) = 0$
- eliminate y and obtain a polynomial $R(x)$ of degree $d = \deg(p) \cdot \deg(q)$
- compute the real zeroes ξ_1, ξ_2, ... of $R(x)$
- analyse the situation at $x = \xi_i$:
  - this amounts to computing the real zeroes of $p(\xi, y)$ and $q(\xi, y)$
  - and to sort them on the line $x = \xi_i$
- computing the real roots of a univariate polynomial is the key subroutine

The Problem

- given a polynomial determine its real roots

- a polynomial is given through its coefficient sequence
- coefficients are real numbers given as potentially infinite bitstreams
- $\pi = 3.14\ldots$
- can ask for additional bits (at no cost)

- determine its real roots = compute isolating intervals for real roots
- an interval $[a,b]$ is isolating if it contains exactly one root of p.

- a measure of difficulty
- $x_1, \ldots, x_n$, the roots of p
- $\mathrm{sep}(p) = \min\{|x_i - x_j| \; ; \; i \neq j\}$, the root separation of p
- intuition: the smaller $\mathrm{sep}(p)$, the harder it is to isolate the roots
- remark: $\mathrm{sep}(p)$ is zero, if p has multiple roots

The Result

- $p(x) = \sum_{0 \le i \le n} p_i x^i$, a polynomial of degree n
- $p_n \ge 1$, $p_i \le 2^\tau$ for all i $\qquad$ τ bits before binary point
- $\operatorname{sep}(p)$, the root separation of p

- **Theorem:** Isolating intervals can be computed in time polynomial in n and $\tau + \log 1/\operatorname{sep}(p)$.

- more precisely, $O(n^4(\tau + \log(1/\operatorname{sep}(p)))^2)$ bit operations
  requires $O(n(\tau + \log(1/\operatorname{sep}(p))))$ bits of each coefficient

- an experiment: $p(x)$, a polynomial with integer coefficients
  running times on $p(x)$, $\pi \cdot p(x)$, and $\sqrt{2} \cdot p(x)$ are essentially the same
- running time depends on geometry of the problem, but not on the representation of the polynomial

Related Work

- well studied problem
- two kinds of papers
  - algorithms without a convergence guarantee
  - algorithms with a guarantee
    - Descartes Method: Descartes, Gauss, Vincent, Uspensky, Ostrowski, Collins/Loos, Krandick/Mehlhorn, Roullier/Zimmermann, Mourrain/Roy/Roullier, ...
    - Root Isolation: Henrici, Schönhage, Pan, Smale, ...
  - Pan has an asymptotically faster algorithm for root isolation
  - but, in his own words:

    The algorithm is quite involved, and would require non-trivial implementation work. No implementation was attempted yet.
  - I doubt that Pan' algorithm would be competitive for our application (computational geometry: many small degree polynomials)

The Bernstein Basis

- it is convenient to work over the so-called Bernstein basis
- interval $[c,d]$, integer n, for $0 \le i \le n$

$$B_i^n(x) = B_i^n[c,d](x) = \binom{n}{i} \frac{(x-c)^i (d-x)^{n-i}}{(d-c)^n}$$

  i-th Bernstein polynomial of degree n with respect to interval $[c,d]$
- The Bernstein polynomials form a basis, i.e., every polynomial p of degree n can be written as

$$p(x) = \sum_{0 \le i \le n} b_i B_i^n[c,d](x)$$

- b_i are called Bernstein coefficients, depend on the interval $[c,d]$.
- $p(c) = b_0$ and $p(d) = b_n$ and $B_i(x) > 0$ for all $x \in (c,d)$

Casteljau's Triangle

- given Bernstein representation with respect to $[c,d]$
- let $m = (c+d)/2$.
- triangle computes reps of $p(x)$ with respect to $[c,m]$ and $[m,d]$
- $b_{j,i} = (b_{j-1,i} + b_{j-1,i+1})/2$ for $j \geq 1$

$$
\begin{array}{ccccccccccc}
b_{0,0} & & b_{0,1} & & b_{0,2} & \ldots & b_{0,n-2} & & b_{0,n-1} & & b_{0,n} \\
& b_{1,0} & & b_{1,1} & & \ldots & & b_{1,n-2} & & b_{1,n-1} & \\
& & b_{2,0} & & b_{2,1} & \ldots & b_{2,n-3} & & b_{2,n-2} & & \\
& & & b_{3,0} & & \ldots & & b_{3,n-3} & & & \\
& & & & . & \ldots & . & & & & \\
& & & & & b_{n,0} & & & & &
\end{array}
$$

- representation for left (right) sub-interval on left (right) side

The Descartes Test

- $b = (b_0, \ldots, b_n)$, sequence of Bernstein coefficients
- number of sign variations $\mathrm{var}(b)$, is the number of pairs (i, j) of integers with $0 \leq i < j \leq n$ and $b_i b_j < 0$ and $b_{i+1} = \ldots = b_{j-1} = 0$

The Descartes Test

- $b = (b_0, \ldots, b_n)$, sequence of Bernstein coefficients
- number of sign variations $\mathrm{var}(b)$, is the number of pairs (i, j) of integers with $0 \le i < j \le n$ and $b_i b_j < 0$ and $b_{i+1} = \ldots = b_{j-1} = 0$
- $\mathrm{var}(-3, 0, -2, 2, -1) = 2$.

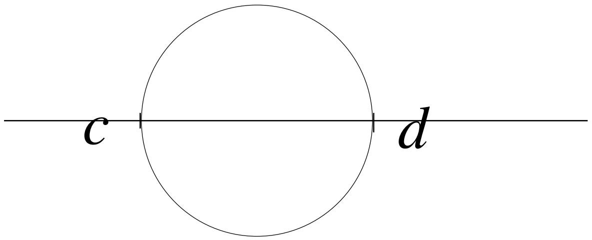
The Descartes Test

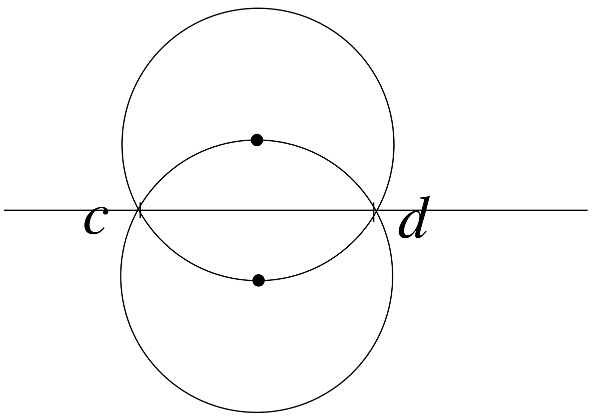
- $b = (b_0, \ldots, b_n)$, sequence of Bernstein coefficients
- number of sign variations $\mathrm{var}(b)$, is the number of pairs (i, j) of integers with $0 \le i < j \le n$ and $b_i b_j < 0$ and $b_{i+1} = \ldots = b_{j-1} = 0$
- $\mathrm{var}(-3, 0, -2, 2, -1) = 2$.
- Theorem (Descartes):
  - number of zeroes of p in (c, d) is at most $\mathrm{var}(b)$
  - both numbers have the same parity
- Corollary:
  - $\mathrm{var}(b) = 0 \quad \Rightarrow \quad$ no root
  - $\mathrm{var}(b) = 1 \quad \Rightarrow \quad$ exactly one root

The Descartes Test

- $b = (b_0, \ldots, b_n)$, sequence of Bernstein coefficients
- number of sign variations $\mathrm{var}(b)$, is the number of pairs (i, j) of integers with $0 \le i < j \le n$ and $b_i b_j < 0$ and $b_{i+1} = \ldots = b_{j-1} = 0$
- $\mathrm{var}(-3, 0, -2, 2, -1) = 2$.
- Theorem (Descartes):
  - number of zeroes of p in (c, d) is at most $\mathrm{var}(b)$
  - both numbers have the same parity
- Corollary:
  - $\mathrm{var}(b) = 0 \quad \Rightarrow \quad$ no root
  - $\mathrm{var}(b) = 1 \quad \Rightarrow \quad$ exactly one root
- Warning: these implications are not iffs, but
- if $\quad d - c \le \mathrm{sep}(p)/4 \quad$ then $\quad \mathrm{var}(b) \le 1$.

The Descartes Test: Partial Converses

- $b = (b_0, \ldots, b_n)$, sequence of Bernstein coefficients
- number of sign variations $\mathrm{var}(b)$, is the number of pairs (i, j) of integers with $0 \le i < j \le n$ and $b_i b_j < 0$ and $b_{i+1} = \ldots = b_{j-1} = 0$
- Landau proves the following partial converses
- if 

contains no root, then $\mathrm{var}(b) = 0$.
- if 

contains exactly one root, then $\mathrm{var}(b) = 1$

A Recursive Algorithm

- task: isolate roots of $p(x)$ in (c,d); $b = (b_0, \ldots, b_n)$ is Bernstein rep
- if $\mathrm{var}(b) = 0$ return;
- if $\mathrm{var}(b) = 1$, return and report $[c,d]$ as an isolating interval
- otherwise. Let $m = (c+d)/2$.
  - Use Casteljau's triangle to compute reps of $p(x)$ wrt sub-intervals
  - If $p(m) = 0$ report $[m,m]$ as an isolating interval. $(p(m) = b_{n,0})$
  - recurse on both sub-intervals.

Analysis

- stopping criteria apply at intervals of length approximately $1/\operatorname{sep}(p)$.
- numbers grow by n bits in every node of the recursion tree (because of the averaging)
- so numbers grow to $L + n\log(M/\operatorname{sep}(p))$ bits where
  - L is the initial length of the numbers (in bits) and
  - M is length of start interval
- for integer coefficients this is the end of the story

Real Coefficients

- need to be able to add coefficients and to determine their sign
- if we start with integer coefficients, this is easy
- but, it is difficult to do for coefficients like π, $\ln 2$, $\sin(\pi/19)$.
- idea: approximate b_i by an interval $[b-\varepsilon, b+\varepsilon]$ with $b_i \in [b-\varepsilon, b+\varepsilon]$
- polynomials become interval polynomials
  - $[a,b]+[c,d]=[a+c,b+d]$
  - here $([a-\varepsilon,a+\varepsilon]+[b-\varepsilon,b+\varepsilon])/2=[(a+b)/2-\varepsilon,(a+b)/2+\varepsilon]$,
  - so we compute with midpoints as described above and simply interpret them as intervals
- Descartes becomes Interval-Descartes (so far, not our idea)

Sign Variations in Sequences of Intervals

- For a sequence of intervals $\mathbf{a} = (\mathbf{a}_0, \ldots, \mathbf{a}_n)$, define its set of potential sign variations: $\operatorname{var}(\mathbf{a}) = \left\{ \operatorname{var}((a_0, \ldots, a_n)) \;\middle|\; a_i \in \mathbf{a}_i \text{ for } 0 \le i \le n \right\}$
-
$$\begin{aligned} \operatorname{var}(([2,3],[-1,1])) &= \{0,1\}\,, \\ \operatorname{var}(([2,3],[-1,1],[2,3])) &= \{0,2\}\,, \\ \operatorname{var}(([2,3],[-1,1],[-2,-1])) &= \{1\}\,. \end{aligned}$$
- The Descartes method hinges crucially on the ability to make the following decisions:
  1. For all nodes in the recursion tree: Does the Descartes test yield 0, 1, or at least 2 sign variations?
  2. For internal nodes of the recursion tree: Does the polynomial vanish at the interval midpoint? (This amounts to testing a certain coefficient for zero, see above.)

  With an interval polynomial, making either decision may be impossible and hence the approach seems doomed.

BUT, RANDOMIZATION MIGHT HELP

- Let $p_0(x)$ be a polynomial having all its roots in $(1/4, 3/4)$.
- choose $\alpha \in [-1/4, +1/4]$ uniformly at random.
- The polynomial $p_\alpha(x) = p_0(x+\alpha)$ has its roots in $(0,1)$.
- Idea: apply the Descartes method to $p = p_\alpha$ and $[0,1]$.

BUT, RANDOMIZATION MIGHT HELP

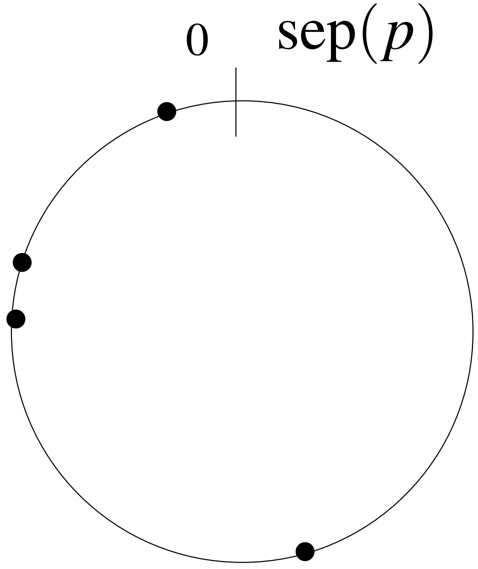
- Let $p_0(x)$ be a polynomial having all its roots in $(1/4, 3/4)$.
- choose $\alpha \in [-1/4, +1/4]$ uniformly at random.
- The polynomial $p_\alpha(x) = p_0(x+\alpha)$ has its roots in $(0,1)$.
- Idea: apply the Descartes method to $p = p_\alpha$ and $[0,1]$.

- a first step: let us argue that $p(x)$ is large for all bisection points x
- this solves the second problem
- bisection points are multiples of $1/2^N$ where $2^{-N} \approx \mathrm{sep}(p)$
- for simplicity assume: $\mathrm{sep}(p)$ is a power of two

More on the Randomization

- we guarantee that our shifted polynomial has value at least $8^n \varepsilon$ at all bisection points (= interval endpoints)
- it is known that a polynomial can be small only close to one of its roots
- make sure that the random shift keeps roots at a reasonable distance from the endpoints of isolating intervals.

    - isolating intervals have length $\mathrm{sep}(p)$
    - a random shift achieves distance $\mathrm{sep}(p)/(4n)$ for all roots with probability at least 1/2



The Smith Bound

- f = polynomial of degree n with roots ξ_1 to ξ_n and $\sigma = \mathrm{sep}(f)$.
- $\tilde{f}$ is f with Bernstein coefficients perturbed by at most ε
- If z is not close to a root, $\tilde{f}(z)$ is large:
  - if $\left|z - \xi_i\right| \geq \dfrac{n(\gamma+\varepsilon)}{\mathrm{lcf}(f)\,\sigma^{n-1}}$ for all i then $\left|\tilde{f}(z)\right| > 8^n \varepsilon$
  - $\mathrm{lcf}(f)$ is the lead coefficient of f (= the coefficient of x^n)

The Smith Bound

- f = polynomial of degree n with roots ξ_1 to ξ_n and $\sigma = \mathrm{sep}(f)$.
- $\tilde{f}$ is f with Bernstein coefficients perturbed by at most ε
- If z is not close to a root, $\tilde{f}(z)$ is large:
  - if $|z - \xi_i| \geq \dfrac{n(\gamma+\varepsilon)}{\mathrm{lcf}(f)\ \sigma^{n-1}}$ for all i then $|\tilde{f}(z)| > 8^n \varepsilon$
  - $\mathrm{lcf}(f)$ is the lead coefficient of f (= the coefficient of x^n)
- randomization guarantees that roots have distance at least $\sigma/(4n)$ from bisection points z, i.e., $|z - \xi_i| \geq \sigma/(4n)$ for all i.
- Thus if

$$\frac{n(8^n \varepsilon + \varepsilon)}{\mathrm{lcf}(f)\ \sigma^{n-1}} \leq \frac{\sigma}{4n} \qquad \text{or} \qquad \varepsilon \leq \frac{\mathrm{lcf}(f)\sigma^n}{n^2 8^n}$$

then $f(z) > 8^n \varepsilon$ for all bisection points

Interval Descartes

- $\tilde{b}$ a vector of intervals of width 2ε each represented by its mid-point.
- $\tilde{b}_i$ is determinate if $|\tilde{b}_i| > \varepsilon$
- $\tilde{b}_i$ is large if $|\tilde{b}_i| > 8^n\varepsilon$ and small otherwise.

- given $\mathbf{p}(x) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(x)$ with $\mathbf{b}_i = [\tilde{b}_i - \varepsilon, \tilde{b}_i + \varepsilon]$.
- Procedure $Descartes_{\text{approx}}$:
  1. If $\tilde{b}_0$ or $\tilde{b}_n$ is small, abort and signal failure.

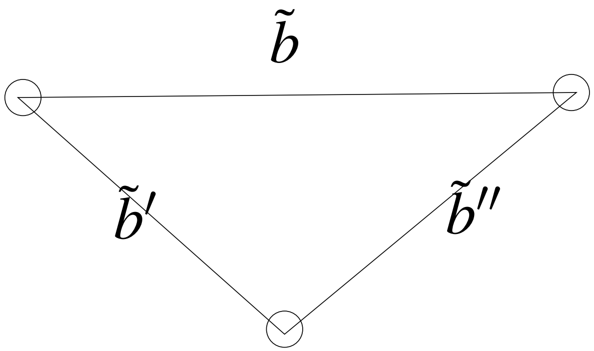
     Otherwise compute $V = \text{var}_\varepsilon(\tilde{b})$, the set of potential sign variations
  2. If $V = \{0\}$, return.
  3. If $V = \{1\}$, report an isolating interval and return.
  4. Compute reps of sub-intervals and recurse.
- We recurse, whenever V contains a value larger than 1.
- If alg fails, $p(x)$ is small at one of the bisection points

  but this is precisely what our randomization avoids

Recursion Depth

- $\tilde{b}$ a vector of intervals of width 2ε each represented by its mid-point.
- $\tilde{b}_i$ is determinate if $|\tilde{b}_i| > \varepsilon$
- $\tilde{b}_i$ is large if $|\tilde{b}_i| > 8^n\varepsilon$ and small otherwise.

- $\tilde{b}'$ and $\tilde{b}''$ are the sequences computed from $\tilde{b}$ by de Casteljau
- $\tilde{b}_0$, $\tilde{b}_n$, $\tilde{b}'_n = \tilde{b}''_0$ are large
- $0 \in \mathrm{var}_\varepsilon(\tilde{b})$ or $1 \in \mathrm{var}_\varepsilon(\tilde{b})$.



- then $\mathrm{var}_\varepsilon(\tilde{b}')$ and $\mathrm{var}_\varepsilon(\tilde{b}'')$ are singleton sets and the recursion stops
- if all extreme coefficients are large, interval Descartes stops one level below exact Descartes

Summary

Summary: If we approximate Berstein coefficients with

$$\log 1/\varepsilon \approx n \log 1/\sigma$$

bits after the binary point, then our algorithm has a constant probability of success when applied to a random shift of p. Recall $\sigma = \mathrm{sep}(p)$.

- can isolate the real roots of arbitrary square-free real polynomials
- even if we cannot compute with the coefficients
- only need to be able to approximate them
- running time is polynomial in the degree and the logarithm of the root separation of the input polynomial
- and this is also the approximation quality needed for the coefficients of the input polynomial

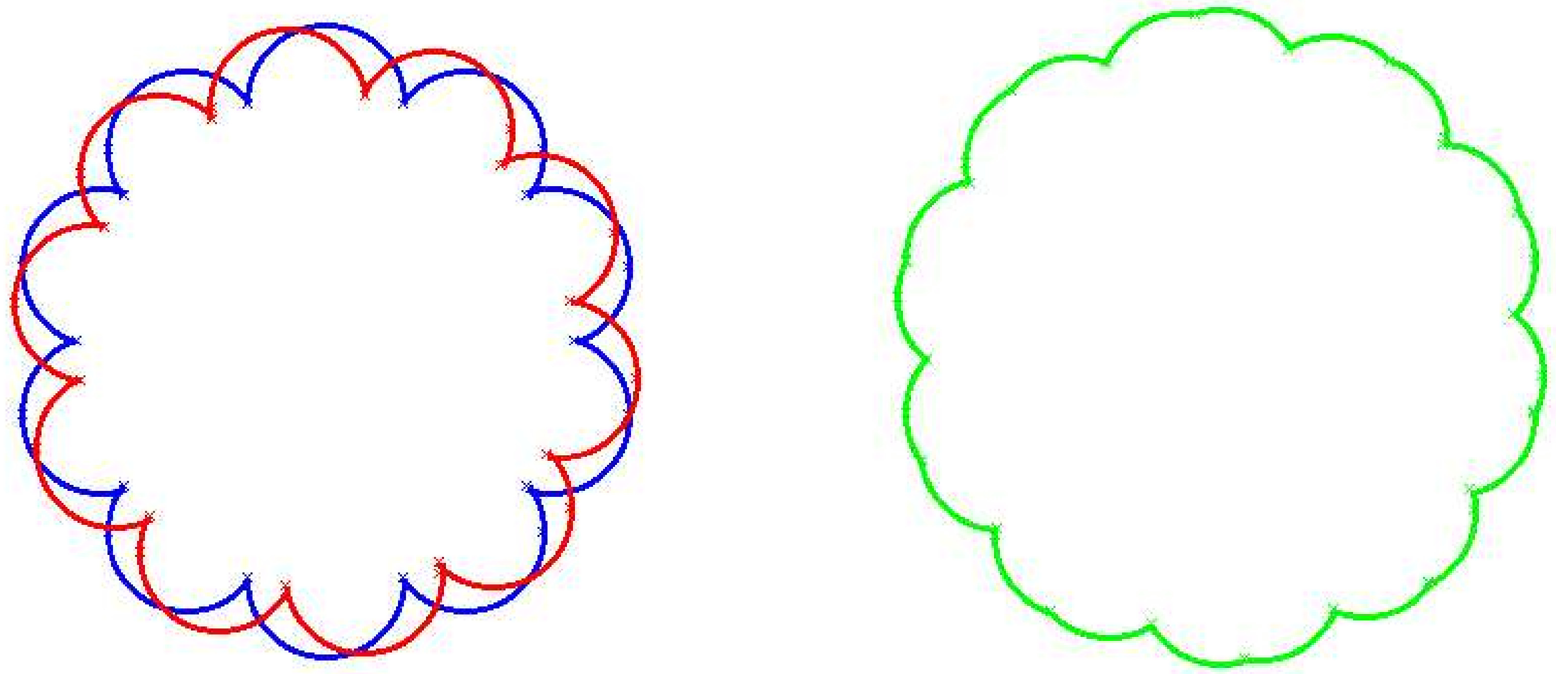
Experiments

- on polynomials with integer coefficients running time of standard Descartes and our version is about the same (give or take a factor of five)

- the big win: running time on $p(x)$ and $\pi \cdot p(x)$ is about the same, i.e.,
- running time depends on the geometry of the problem (distribution of roots in the plane) and not on the idiosyncrasy of the representation

- alg has improved our geometry implementations significantly

Curved Polygons



green polygon is union of red and blue

computation time decreased by factor 1000 over the past three years

it now below one second for polygons with 1000 vertices and edges