Computing Real Roots of Real Polynomials ...
... and now For Real!

Alexander Kobel · Max-Planck-Institute for Informatics, Saarbrücken, Germany
Fabrice Rouillier · INRIA & Université Pierre et Marie Curie, Paris, France
Michael Sagraloff · Max-Planck-Institute for Informatics, Saarbrücken, Germany



Computing Real Roots ...

A classical approach

Computing Real Roots of Real Polynomials

Given a square-free polynomial

$$P(x) = p_n x^n + p_{n-1} x^{n-1} + \cdots + p_1 x + p_0 \in \mathbb{R}[x]$$

and an open interval $\mathcal{I} = (a_0, b_0) \subset \mathbb{R}$,

find *isolating intervals* $I_1, \ldots, I_r \subset \mathcal{I}$ such that

- each I_j contains exactly one real root of P,
- the I_j are pairwise disjoint, and
- $\bigcup_j I_j$ covers all real roots of P in $\mathcal{I}$.

Subdivision approach

- *Queue* $\leftarrow \{\mathcal{I}\}$
  Isol $\leftarrow \emptyset$
- while *Queue* is not empty:
  - pop $I = (a, b)$ from *Queue*
  - if I contains no root: *(exclusion predicate)*
    discard I
  - else if I contains exactly one root: *(inclusion-and-isolation predicate)*
    add I to *Isol*
  - otherwise (or we-don't-know):
    - split I at $m = \frac{a+b}{2}$ *(or some other point)*
    - if $P(m) = 0$: add $\{m\}$ to *Isol*
    - add (a, m) and (m, b) to *Queue*

Subdivision approach

- *Queue* $\leftarrow \{\mathcal{I}\}$
  Isol $\leftarrow \emptyset$
- while *Queue* is not empty:
  - pop $I = (a, b)$ from *Queue*
  - if I contains no root: *(exclusion predicate)*
    discard I
  - else if I contains exactly one root: *(inclusion-and-isolation predicate)*
    add I to *Isol*
  - otherwise (or we-don't-know):
    - split I at $m = \frac{a+b}{2}$ *(or some other point)*
    - if $P(m) = 0$: add $\{m\}$ to *Isol*
    - add (a, m) and (m, b) to *Queue*

Descartes' Rule of Signs

$r =$ # positive real roots of P

$v =$ # sign changes in coeffs of P

$$r \leq v \text{ and } r \equiv v \pmod 2$$

Descartes' Rule of Signs

$r =$ # positive real roots of P

$v =$ # sign changes in coeffs of P

$$r \leq v \text{ and } r \equiv v \pmod 2$$

localized version

$r_I =$ # real roots of P in $I = (a, b)$

$v_I =$ # sign changes of $(x+1)^n P\left(\frac{ax+b}{x+1}\right)$

$$r_I \leq v_I \text{ and } r_I \equiv v_I \pmod 2$$

Descartes' Rule of Signs

$r =$ # positive real roots of P

$v =$ # sign changes in coeffs of P

$$r \leq v \text{ and } r \equiv v \pmod 2$$

localized version

$r_I =$ # real roots of P in $I = (a, b)$

$v_I =$ # sign changes of $(x+1)^n P\left(\frac{ax+b}{x+1}\right)$

$$r_I \leq v_I \text{ and } r_I \equiv v_I \pmod 2$$

exclusion predicate

$v_I = 0 \quad \Rightarrow \quad I$ contains no root of P

inclusion-and-isolation predicate

$v_I = 1 \quad \Rightarrow \quad I$ contains exactly one root of P

Descartes' Rule of Signs

$r = \#$ positive real roots of P

$v = \#$ sign changes in coeffs of P

$$r \leq v \text{ and } r \equiv v \pmod 2$$

localized version

$r_I = \#$ real roots of P in $I = (a, b)$

$v_I = \#$ sign changes of $(x+1)^n P\left(\frac{ax+b}{x+1}\right)$

$$r_I \leq v_I \text{ and } r_I \equiv v_I \pmod 2$$

exclusion predicate

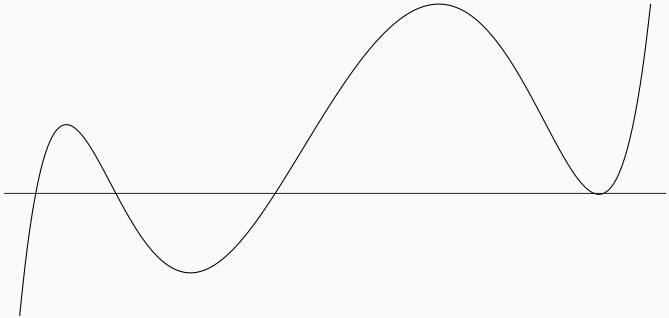
$v_I = 0 \quad \Rightarrow \quad I$ contains no root of P

inclusion-and-isolation predicate

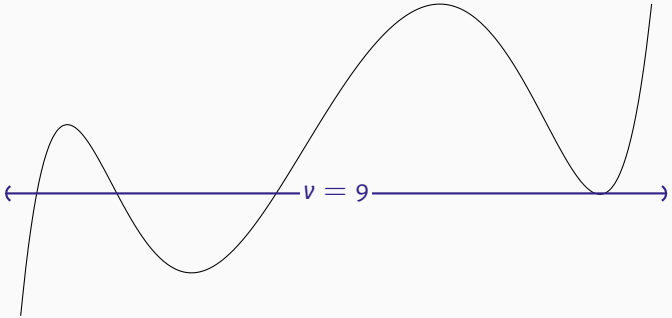
$v_I = 1 \quad \Rightarrow \quad I$ contains exactly one root of P

$r_I = v_I \in \{0, 1\}$ if other (complex) roots well-separated from I

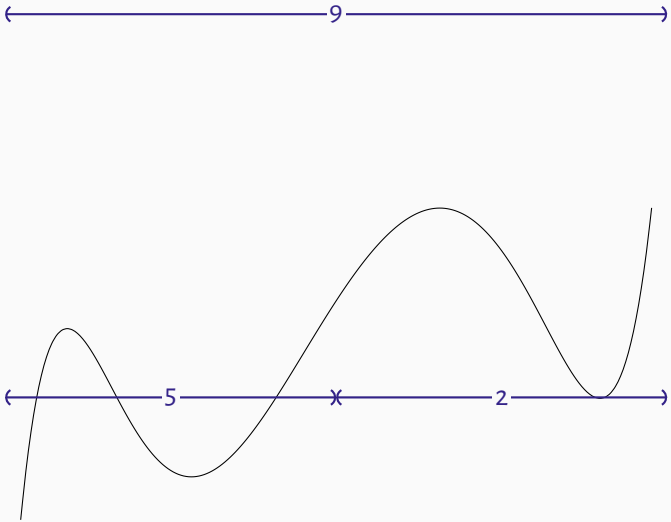
Descartes method



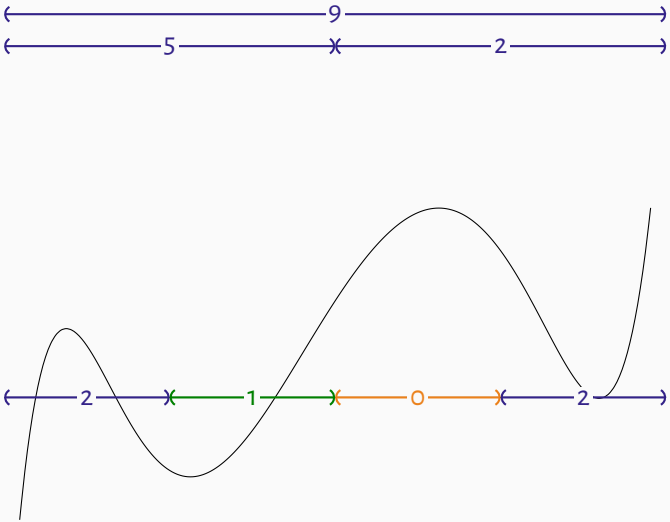
Descartes method



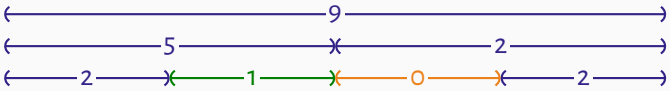
Descartes method

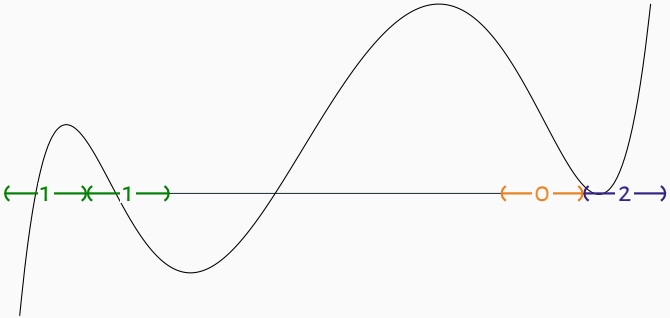


Descartes method

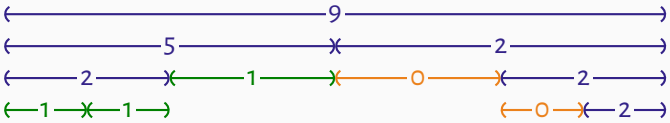


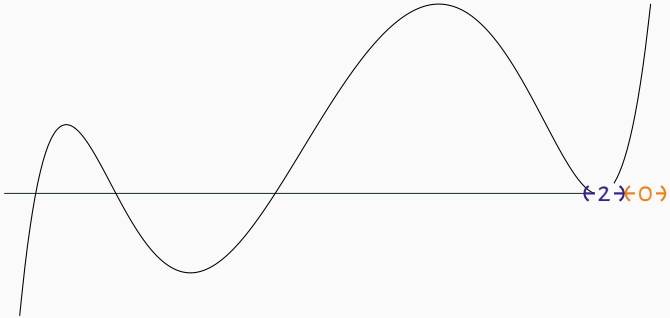
Descartes method



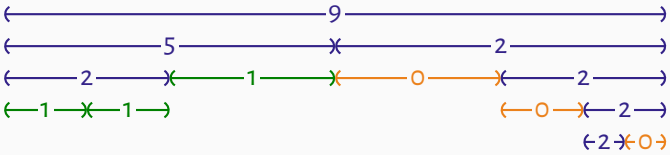


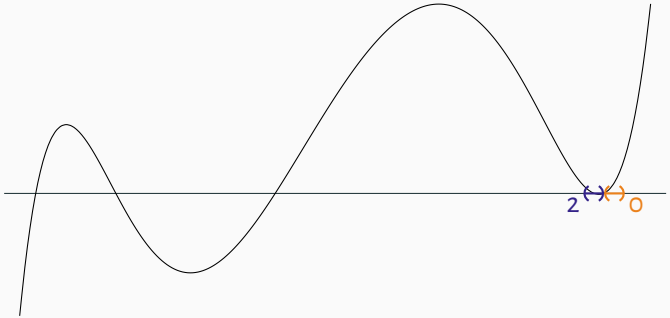
Descartes method



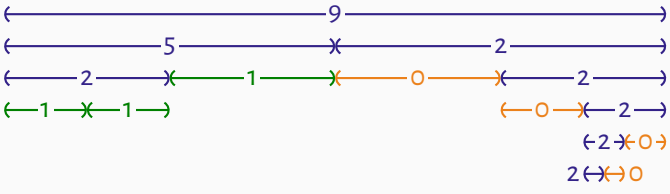


Descartes method

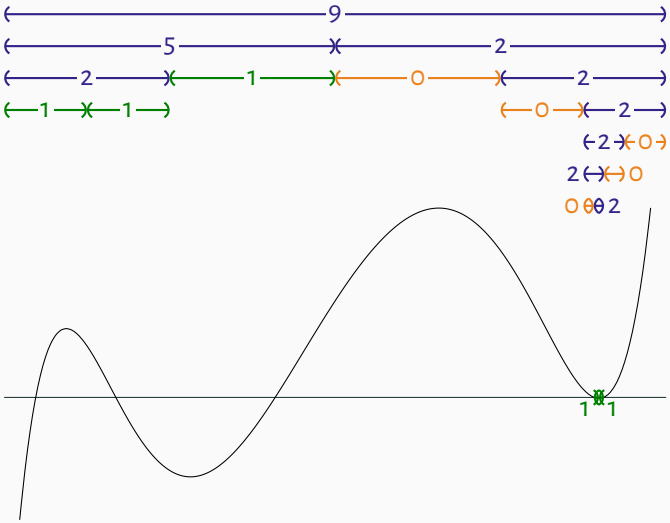




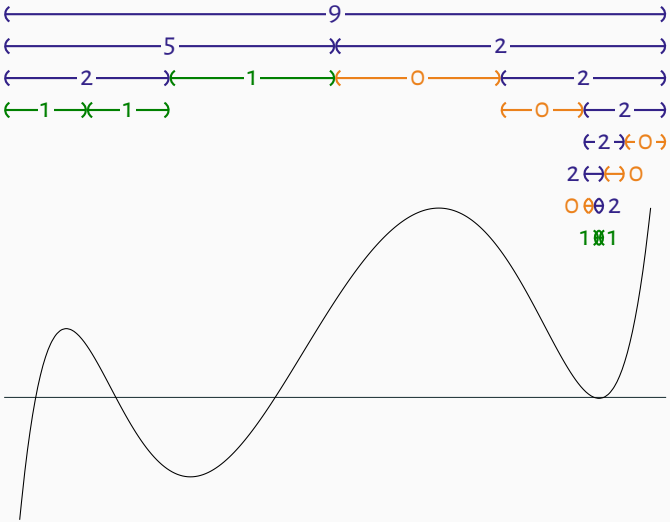
Descartes method



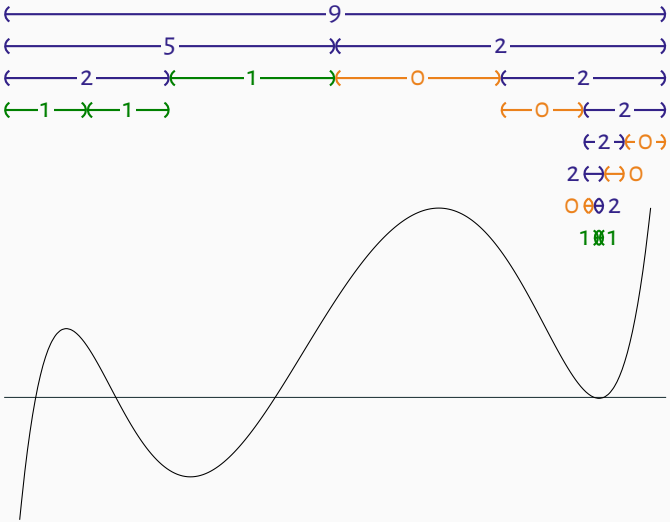
Descartes method

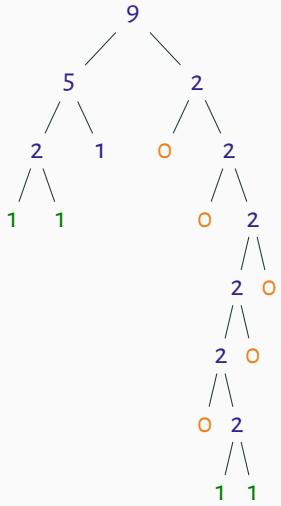


Descartes method



Descartes method





Complexity ("benchmark problem")

isolate *all* roots of an *integer* polynomial of degree n and coefficient bitsize τ

roots / tree width: $\leq n$ min. root separation: $\geq 2^{-\mathcal{O}(n\tau)}$

isolate *all* roots of an *integer* polynomial of degree n and coefficient bitsize τ

roots / tree width: $\leq n$ min. root separation: $\geq 2^{-\mathcal{O}(n\tau)}$

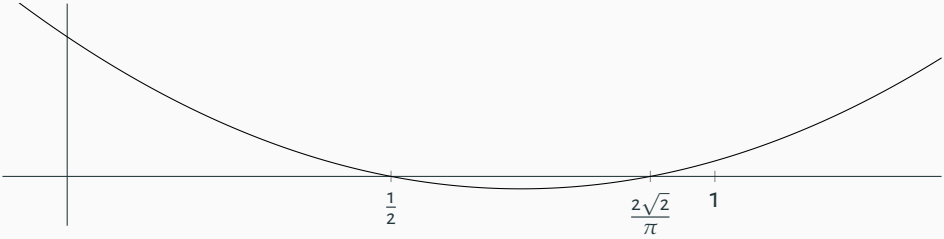
subdivision rule and model of computation	subdivision tree size	precision demand	overall bit complexity
bisection, exact over $\mathbb{Z}$ / $\mathbb{Q}$	$\tilde{\mathcal{O}}(n\tau)$[1]	$\tilde{\mathcal{O}}(n^2\tau)$	$\tilde{\mathcal{O}}(n^4\tau^2)$[1]

[1] [Eigenwillig, Sharma, Yap: ISSAC 2006] [Collins: JSC 2016]

… of Real Polynomials …

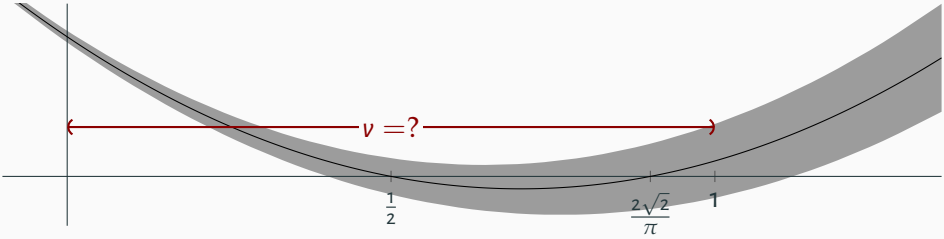
Deficiencies and remedies

$$P(x) = 2\pi x^2 - (\pi + 4\sqrt{2})x + 2\sqrt{2} = (\pi x - 2\sqrt{2})(2x - 1) \quad \in \mathbb{R}[x]$$
$$\approx 6.2831853072x^2 - 8.7984469031x + 2.8284271247$$



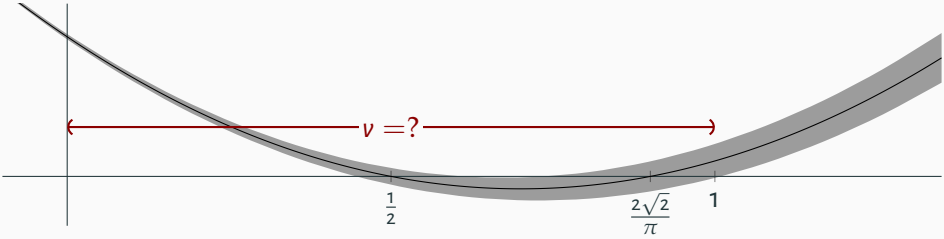
$$P(x) = 2\pi x^2 - (\pi + 4\sqrt{2})x + 2\sqrt{2} = (\pi x - 2\sqrt{2})(2x - 1) \quad \in \mathbb{R}[x]$$

$$\approx \qquad 6....x^2 - \qquad 9....x + \qquad 3....$$



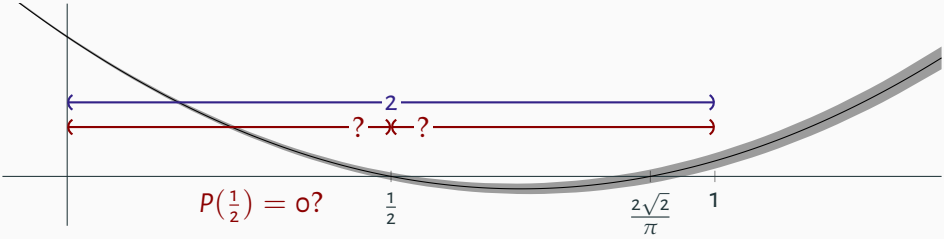
$$P(x) = 2\pi x^2 - (\pi + 4\sqrt{2})x + 2\sqrt{2} = (\pi x - 2\sqrt{2})(2x - 1) \quad \in \mathbb{R}[x]$$

$$\approx \quad 6.3\ldots x^2 - \quad 8.8\ldots x + \quad 2.8\ldots$$



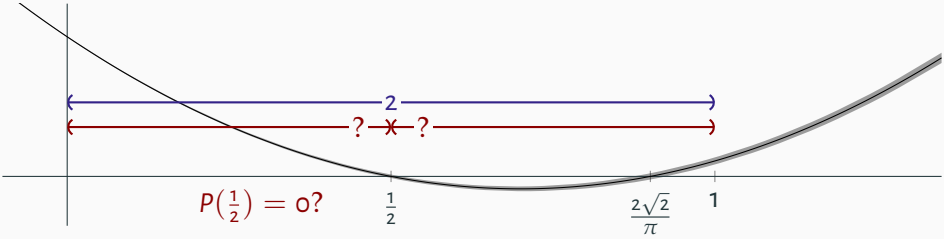
$$P(x) = 2\pi x^2 - (\pi + 4\sqrt{2})x + 2\sqrt{2} = (\pi x - 2\sqrt{2})(2x - 1) \quad \in \mathbb{R}[x]$$

$$\approx \quad 6.28\ldots x^2 - \quad 8.79\ldots x + \quad 2.83\ldots$$



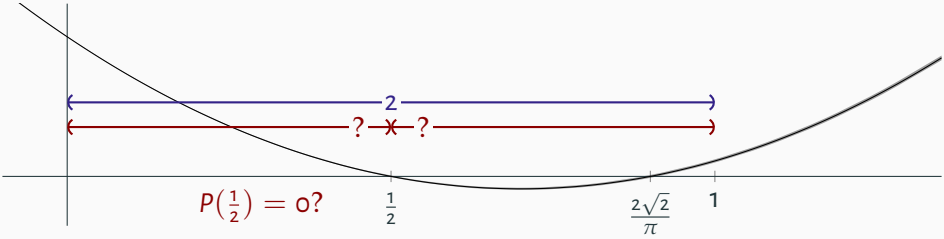
$$P(x) = 2\pi x^2 - (\pi + 4\sqrt{2})x + 2\sqrt{2} = (\pi x - 2\sqrt{2})(2x - 1) \quad \in \mathbb{R}[x]$$

$$\approx \quad 6.283\ldots x^2 - \quad 8.7984\ldots x + \quad 2.828\ldots$$

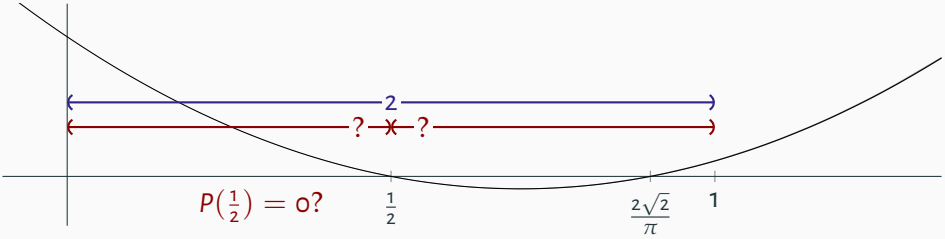


$$P(x) = 2\pi x^2 - (\pi + 4\sqrt{2})x + 2\sqrt{2} = (\pi x - 2\sqrt{2})(2x - 1) \quad \in \mathbb{R}[x]$$

$$\approx \quad 6.2832\ldots x^2 - \quad 8.79845\ldots x + \quad 2.8284\ldots$$



$$P(x) = 2\pi x^2 - (\pi + 4\sqrt{2})x + 2\sqrt{2} = (\pi x - 2\sqrt{2})(2x - 1) \quad \in \mathbb{R}[x]$$
$$\approx \quad 6.2832\ldots x^2 - \quad 8.79845\ldots x + \quad 2.8284\ldots$$



problems with arbitrarily approximable ("bitstream") coefficients

- *no termination* on subdivision on an exact root
- *precision demand* for processing $I = (a, b)$ depends on $|P(a)|$ and $|P(b)|$

ADsc (Approximate Descartes)

Instead of subdivision at $m \in I$:

- sample suitable points near m,
- choose an *admissible point* m^* with large value $|P(m^*)|$ among the samples
- and subdivide at m^*.

ADsc (Approximate Descartes)

Instead of subdivision at $m \in I$:

- sample suitable points near m,
- choose an *admissible point* m^* with large value $|P(m^*)|$ among the samples
- and subdivide at m^*.

In particular: $|P(m^*)| \neq 0$!

$\Rightarrow$ *guaranteed termination* for arbitrary (square-free) inputs

$\Rightarrow$ keeps *precision demand near theoretical optimum*

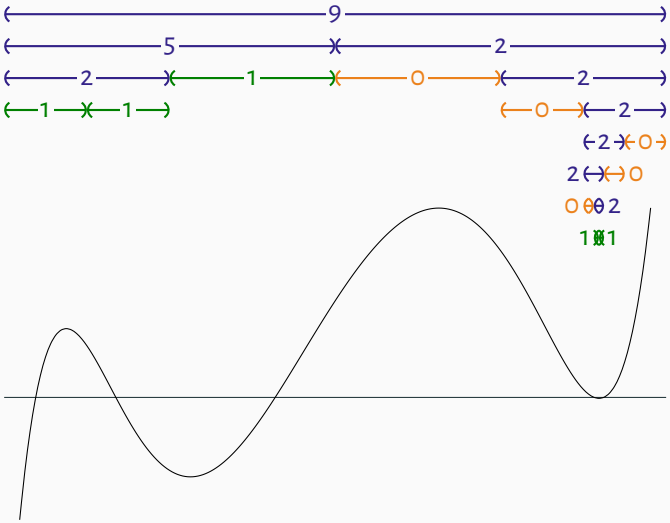
isolate *all* roots of an *integer* polynomial of degree n and coefficient bitsize τ

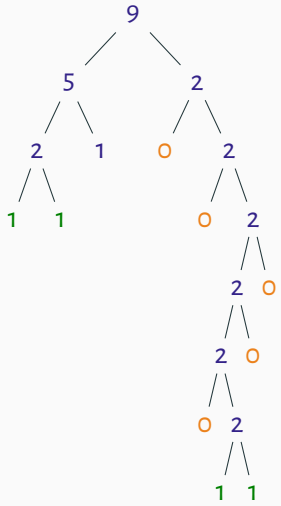
roots / tree width: $\leq n$ min. root separation: $\geq 2^{-\mathcal{O}(n\tau)}$

subdivision rule and model of computation	subdivision tree size	precision demand	overall bit complexity
bisection, exact over $\mathbb{Z}$ / $\mathbb{Q}$	$\tilde{\mathcal{O}}(n\tau)$	$\tilde{\mathcal{O}}(n^2\tau)$	$\tilde{\mathcal{O}}(n^4\tau^2)$
bisection, approximate	$\tilde{\mathcal{O}}(n\tau)$	$\tilde{\mathcal{O}}(n\tau)$[1]	$\tilde{\mathcal{O}}(n^3\tau^2)$[1]

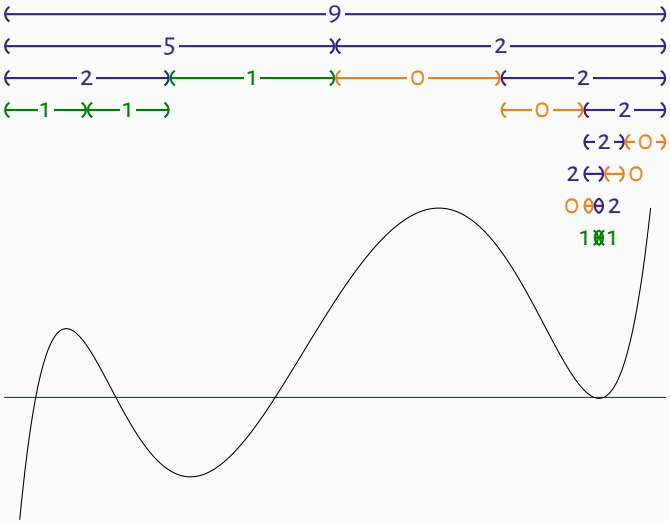
[1] [Sagraloff: JSC 2014]

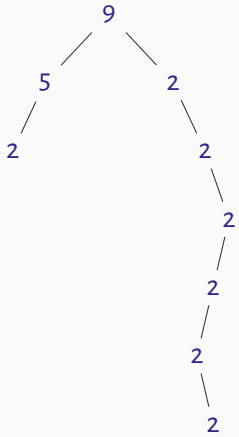
Clustered roots



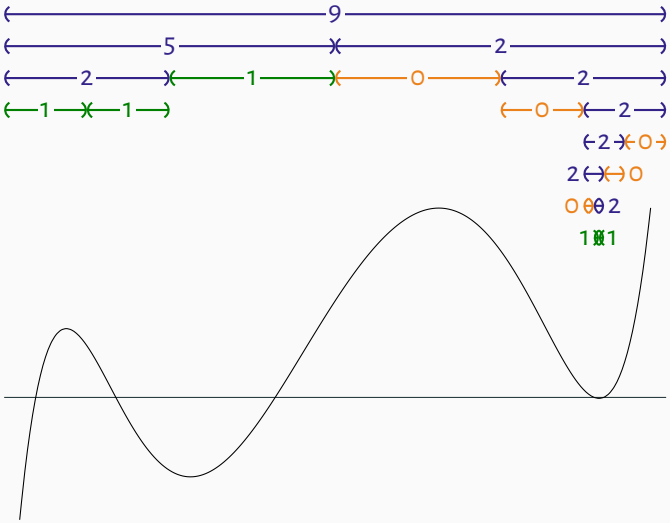


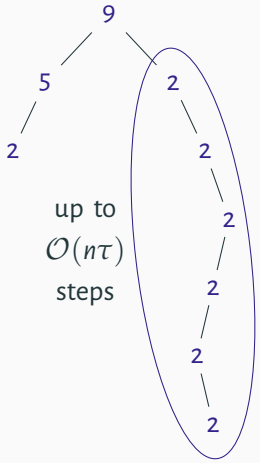
Clustered roots





Clustered roots





Combining Newton's and Descartes method

- inspired by (Approximate) Quadratic Interval Refinement[2] and the Brent-Dekker method[3]

[2] [Abbott: ISSAC 2006; ACM CCA 2014], [Kerber, Sagraloff: JCAM 2015]

[3] [Brent: 1973]

Combining Newton's and Descartes method

- inspired by (Approximate) Quadratic Interval Refinement[2] and the Brent-Dekker method[3]
- certified trial-and-error approach:
  - multiplicity-agnostic variant of *Newton iteration*
  - determine *candidate subinterval* of I containing a cluster
  - *verify* using Descartes tests; on failure, resort to bisection

[2] [Abbott: ISSAC 2006; ACM CCA 2014], [Kerber, Sagraloff: JCAM 2015]

[3] [Brent: 1973]

Combining Newton's and Descartes method

- inspired by (Approximate) Quadratic Interval Refinement[2]
  and the Brent-Dekker method[3]
- certified trial-and-error approach:
  - multiplicity-agnostic variant of *Newton iteration*
  - determine *candidate subinterval* of I containing a cluster
  - *verify* using Descartes tests; on failure, resort to bisection
- successful in almost all steps;
  reduces chains of subdivisions around a cluster to poly-logarithmic length

[2] [Abbott: ISSAC 2006; ACM CCA 2014], [Kerber, Sagraloff: JCAM 2015]

[3] [Brent: 1973]

Combining Newton's and Descartes method

- inspired by (Approximate) Quadratic Interval Refinement[2]
  and the Brent-Dekker method[3]
- certified trial-and-error approach:
  - multiplicity-agnostic variant of *Newton iteration*
  - determine *candidate subinterval* of I containing a cluster
  - *verify* using Descartes tests; on failure, resort to bisection
- successful in almost all steps;
  reduces chains of subdivisions around a cluster to poly-logarithmic length
- similar techniques in complex domain: upcoming talk by Juan Xu

[2] [Abbott: ISSAC 2006; ACM CCA 2014], [Kerber, Sagraloff: JCAM 2015]

[3] [Brent: 1973]

Complexity ("benchmark problem")

isolate *all* roots of an *integer* polynomial of degree n and coefficient bitsize τ

roots / tree width: $\leq n$ min. root separation: $\geq 2^{-\mathcal{O}(n\tau)}$

subdivision rule and model of computation	subdivision tree size	precision demand	overall bit complexity
bisection, exact over $\mathbb{Z}$ / $\mathbb{Q}$	$\tilde{\mathcal{O}}(n\tau)$	$\tilde{\mathcal{O}}(n^2\tau)$	$\tilde{\mathcal{O}}(n^4\tau^2)$
bisection, approximate	$\tilde{\mathcal{O}}(n\tau)$	$\tilde{\mathcal{O}}(n\tau)$	$\tilde{\mathcal{O}}(n^3\tau^2)$
Newton, exact over $\mathbb{Z}$ / $\mathbb{Q}$	$\tilde{\mathcal{O}}(n)$[1]	$\tilde{\mathcal{O}}(n^2\tau)^*$	$\tilde{\mathcal{O}}(n^3\tau)$[1]

*amortized over entire tree: $\tilde{\mathcal{O}}(n\tau)$

[1] [Sagraloff: ISSAC 2012]

isolate *all* roots of an *integer* polynomial of degree n and coefficient bitsize τ

roots / tree width: $\leq n$ min. root separation: $\geq 2^{-\mathcal{O}(n\tau)}$

subdivision rule and model of computation	subdivision tree size	precision demand	overall bit complexity
bisection, exact over $\mathbb{Z}$ / $\mathbb{Q}$	$\tilde{\mathcal{O}}(n\tau)$	$\tilde{\mathcal{O}}(n^2\tau)$	$\tilde{\mathcal{O}}(n^4\tau^2)$
bisection, approximate	$\tilde{\mathcal{O}}(n\tau)$	$\tilde{\mathcal{O}}(n\tau)$	$\tilde{\mathcal{O}}(n^3\tau^2)$
Newton, exact over $\mathbb{Z}$ / $\mathbb{Q}$	$\tilde{\mathcal{O}}(n)$	$\tilde{\mathcal{O}}(n^2\tau)^*$	$\tilde{\mathcal{O}}(n^3\tau)$
Newton, approximate	$\tilde{\mathcal{O}}(n)$	$\tilde{\mathcal{O}}(n\tau)^\dagger$	$\tilde{\mathcal{O}}(n^3 + n^2\tau)$[1]

*amortized over entire tree: $\tilde{\mathcal{O}}(n\tau)$

†amortized over entire tree: $\tilde{\mathcal{O}}(n + \tau)$

[1] [Sagraloff, Mehlhorn: JSC 2016]
best known: $\tilde{\mathcal{O}}(n^2\tau)$ [Pan: JSC 2002] [Mehlhorn, Sagraloff, Wang: JSC 2015]

… and now For Real!

Implementation results

RS

- C library for real root solving, refinement & more
- sophisticated implementation of classical Descartes
- tailored for approximate arithmetic
- high-performance general purpose solver
- default real root solver in Maple since version 11

[Rouillier, Zimmermann: JCAM 2003]

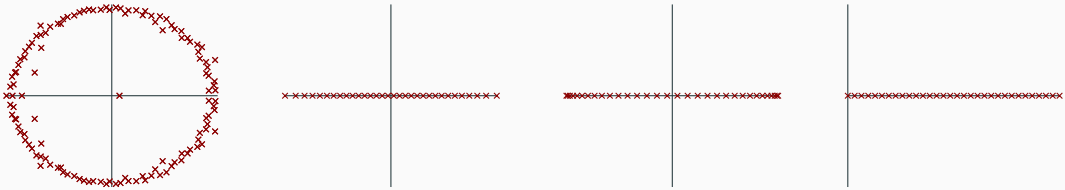
ANewDsc (Approximate Arithmetic Newton-Descartes)

- implemented on top of RS
- merged admissible point selection & Newton-Descartes
- heuristics to reduce / eliminate overhead on “easy” instances
- additional optimizations
- certified output (based on interval arithmetic using MPFI)
- matches theoretical worst-case bit complexity (Las Vegas)
  (assuming asymptotically fast polynomial arithmetic)
- to be integrated in Maple 201x?

[Sagraloff, Mehlhorn: JSC 2016]

Benchmark excerpt: tame (well-separated) instances

instance	degree	bitsize	RS		ANewDsc		speedup
			time [s]	nodes	time [s]	nodes	
random	16384	1024	275.1	46	279.0	40	0.99
Hermite	1024	9875	88.3	2237	79.6	1396	1.11
Legendre	1024	4640	118.2	2272	117.2	1325	1.01
Wilkinson	1024	8777	219.1	2064	398.8	1565	0.73

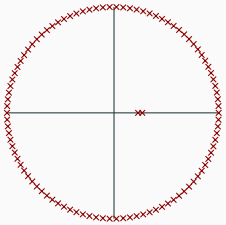


Benchmark excerpt: hard (clustered) instances

instance	degree	bitsize	RS		ANewDsc		speedup
			time [s]	nodes	time [s]	nodes	
monic random	4096	1024	713.1	3212	14.6	37	48.84
random$^2 - 1$	1024	1029	288.5	6648	5.2	419	55.48

Benchmark excerpt: hard (clustered) instances

instance	degree	bitsize	RS		ANewDsc		speedup
			time [s]	nodes	time [s]	nodes	
Mignotte polynomials: $x^n - (ax-1)^2$							
$a = 2^{31}$	1025	63	3.6	66	1.4	27	2.57
$a = 5$	1025	5	402.6	2384	0.7	43	575.14
$a = 2^{31} + 1$	1025	63	13338.4	31810	1.2	53	11115.3

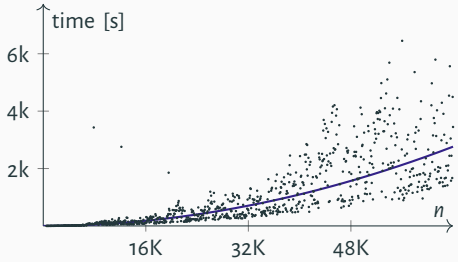


two clustered roots near $1/a$
with separation of appx. $a^{n/2}$

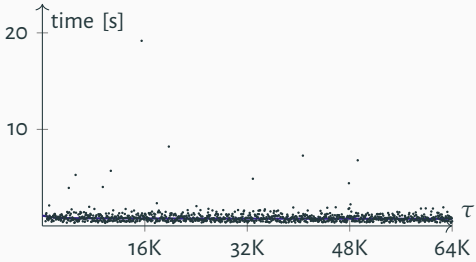
Comparison to other solvers

instance	n	τ	MPSolve	CF	Sage	RS	ANewDsc
random	16384	1024	578.9	376.6	3825.1	275.1	279.0
Hermite	1024	9875	2169.9	140.5	11.9	88.3	79.6
Legendre	1024	4640	5061.3	79.0	9.7	118.2	117.2
Wilkinson	1024	8777	5308.1	13.7	14.1	219.1	398.8
random$^2 - 1$	512	1029	33.5	339.2	3.1	71.5	1.3
	1024	1029	141.8	> 7200	8.2	288.5	5.2
random monic	1024	1024	2.6	0.7	4115.0	38.8	1.2
	16384	1024	579.3	279.2	> 7200	> 7200	143.1
$x^n - (2^a x - 1)^2$	2047	5	87.2	1.2	21.1	1.0	1.2
$x^n - (ax - 1)^2$	2047	5	131.5	1.5	21.7	1.6	1.6
$x^n - (ax^2 - 1)^2$	2047	5	99.1	> 7200	26.5	3472.0	8.3
nested 4-fold	260	2560	69.6	348.8	5.6	636.7	2.1
	1028	260	186.2	3993.7	20.9	430.4	7.4

Bit complexity estimation: tame instances



estimated exponent for degree: 1.97



estimated exponent for bitsize: -0.03

theoretical vs. estimated bit complexity:

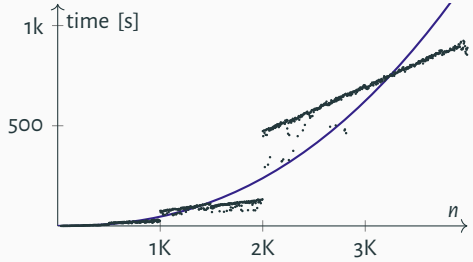
$$\tilde{\mathcal{O}}(n^3 + n^2\tau) \text{ vs. } \mathcal{O}(n^{1.97}\tau^{-0.03})$$

---

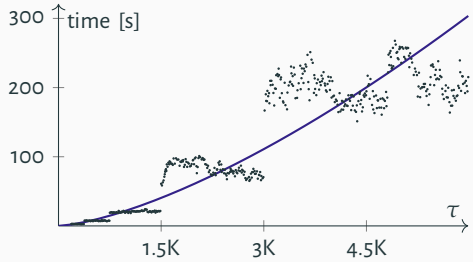
degree n, integer coefficients in range $(-2^\tau, 2^\tau)$ chosen uniformly at random

left: $\tau = 1024$; right: $n = 1024$

Bit complexity estimation: hard instances



estimated exponent for degree: 2.37



estimated exponent for bitsize: 1.45

theoretical vs. estimated bit complexity:

$$\tilde{\mathcal{O}}(n^3 + n^2\tau) \text{ vs. } \mathcal{O}(n^{2.37}\tau^{1.45})$$

Mignotte-like: $x^n - (ax^2 - 1)^3$ (cluster of multiplicity 3 around irrational center)
left: $a = 2^{256} - 1$; right: $n = 1024$

ANewDsc: Approximate Arithmetic Newton-Descartes

- highly efficient general-purpose solver

ANewDsc: Approximate Arithmetic Newton-Descartes

- highly efficient general-purpose solver
- tremendous speedups in degenerate situations
  without tailored tweaks (e.g., does not (yet) exploit sparsity)

ANewDsc: Approximate Arithmetic Newton-Descartes

- highly efficient general-purpose solver
- tremendous speedups in degenerate situations
  without tailored tweaks (e.g., does not (yet) exploit sparsity)

no 

ANewDsc: Approximate Arithmetic Newton-Descartes

- highly efficient general-purpose solver
- tremendous speedups in degenerate situations
  without tailored tweaks (e.g., does not (yet) exploit sparsity)
- easily solves previously infeasible instances

no 

ANewDsc: Approximate Arithmetic Newton-Descartes

- highly efficient general-purpose solver
- tremendous speedups in degenerate situations
  without tailored tweaks (e.g., does not (yet) exploit sparsity)
- easily solves previously infeasible instances
- first available solver with
  - expected performance near theoretical optimum
  - native support for inputs with arbitrary real coefficients

no 

ANewDsc: Approximate Arithmetic Newton-Descartes

- highly efficient general-purpose solver
- tremendous speedups in degenerate situations
  without tailored tweaks (e.g., does not (yet) exploit sparsity)
- easily solves previously infeasible instances
- first available solver with
  - expected performance near theoretical optimum
  - native support for inputs with arbitrary real coefficients
- implementation & benchmarks available at ANewDsc.mpi-inf.mpg.de

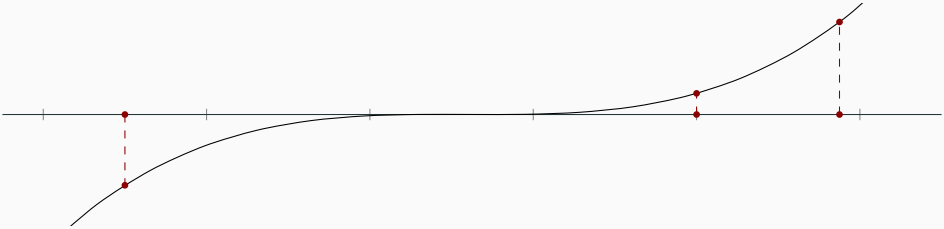
no 

- highly efficient general-purpose solver
- tremendous speedups in degenerate situations
  without tailored tweaks (e.g., does not (yet) exploit sparsity)
- easily solves previously infeasible instances
- first available solver with
  - expected performance near theoretical optimum
  - native support for inputs with arbitrary real coefficients
- implementation & benchmarks available at ANewDsc.mpi-inf.mpg.de

no 

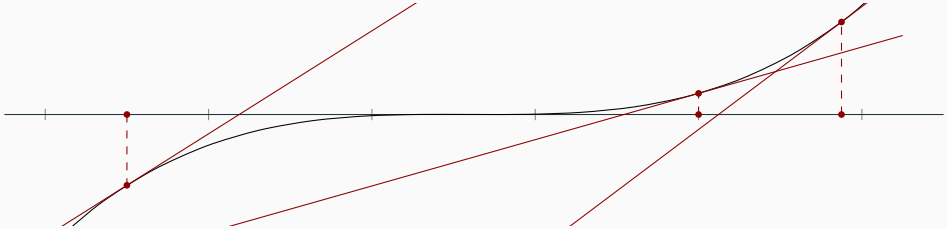
Thank you for your attention!

Newton-Descartes



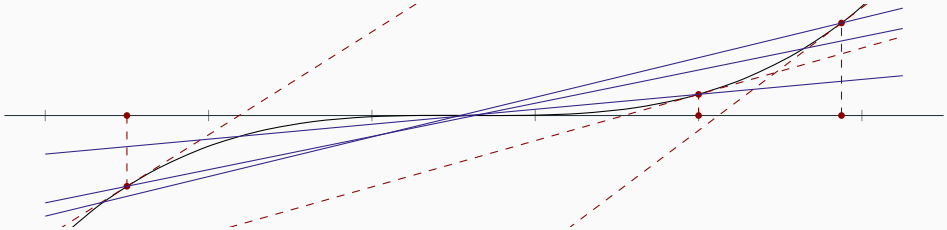
- choose samples x_i

Newton-Descartes



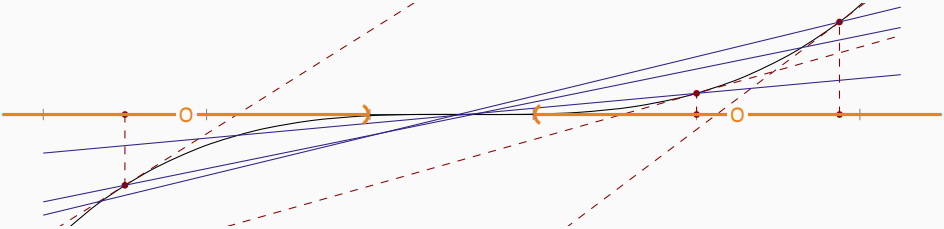
- choose samples x_i
- compute Newton correction terms $N(x_i) = P(x_i)/P'(x_i)$

Newton-Descartes



- choose samples x_i
- compute Newton correction terms $N(x_i) = P(x_i)/P'(x_i)$
- guess multiplicity k s.t. $x_i - k \cdot N(x_i)$ is approximately the same

Newton-Descartes



- choose samples x_i
- compute Newton correction terms $N(x_i) = P(x_i)/P'(x_i)$
- guess multiplicity k s.t. $x_i - k \cdot N(x_i)$ is approximately the same
- verify that no roots are lost; otherwise, resort to bisection

Comparison to other solvers

instance	n	τ	MPSolve	CF	Sage	SLV	RS	ANewDsc
random	16384	1024	578.9	376.6	3825.1	5138.1	275.1	279.0
Hermite	1024	9875	2169.9	140.5	11.9	176.3	88.3	79.6
Legendre	1024	4640	5061.3	79.0	9.7	215.2	118.2	117.2
Wilkinson	1024	8777	5308.1	13.7	14.1	(80.9)	219.1	398.8
random$^2 - 1$	512	1029	33.5	339.2	3.1	3270.5	71.5	1.3
	1024	1029	141.8	> 7200	8.2	> 7200	288.5	5.2
random monic	1024	1024	2.6	0.7	4115.0	> 7200	38.8	1.2
	16384	1024	579.3	279.2	> 7200	> 7200	> 7200	143.1
$x^n - (2^a x - 1)^2$	2047	5	87.2	1.2	21.1	1.1	1.0	1.2
$x^n - (ax - 1)^2$	2047	5	131.5	1.5	21.7	0.7	1.6	1.6
$x^n - (ax^2 - 1)^2$	2047	5	99.1	> 7200	26.5	> 7200	3472.0	8.3
nested 4-fold	260	2560	69.6	348.8	5.6	1193.9	636.7	2.1
	1028	260	186.2	3993.7	20.9	6983.4	430.4	7.4