Computing real roots of real polynomials ... ... and now for real!

Alexander Kobel
Max-Planck-Institut für Informatik
Universität des Saarlandes
Campus E1 4
66123 Saarbrücken, Germany
alexander.kobel@mpi-inf.mpg.de

Fabrice Rouillier
INRIA Paris
Institut de Mathématiques de Jussieu Paris Rive Gauche
CNRS UMR 7586
Université P. & M. Curie
4 place Jussieu 75252 Paris Cedex 05, France
Fabrice.Rouillier@inria.fr

Michael Sagraloff
Max-Planck-Institut für Informatik
Campus E1 4
66123 Saarbrücken, Germany
michael.sagraloff@mpi-inf.mpg.de

ABSTRACT

Very recent work introduces an asymptotically fast subdivision algorithm, denoted ANewDsc, for isolating the real roots of a univariate real polynomial. The method combines Descartes' Rule of Signs to test intervals for the existence of roots, Newton iteration to speed up convergence against clusters of roots, and approximate computation to decrease the required precision. It achieves record bounds on the worst-case complexity for the considered problem, matching the complexity of Pan's method for computing all complex roots and improving upon the complexity of other subdivision methods by several magnitudes.

In the article at hand, we report on an implementation of ANewDsc on top of the RS root isolator. RS is a highly efficient realization of the classical Descartes method and currently serves as the default real root solver in Maple. We describe crucial design changes within ANewDsc and RS that led to a high-performance implementation without harming the theoretical complexity of the underlying algorithm.

With an excerpt of our collection of benchmarks, available at http://anewdsc.mpi-inf.mpg.de/, we illustrate that the theoretical gain in performance of ANewDsc over other subdivision methods also transfers into practice. These experiments also show that our new implementation outperforms both RS and mature competitors by magnitudes for notoriously hard instances with clustered roots. For all other instances, we avoid almost any overhead by integrating additional optimizations and heuristics.

Keywords

real roots; univariate polynomials; root finding; root isolation; Newton's method; Descartes method; approximate arithmetic; certified computation

Accepted for presentation at the 41st International Symposium on Symbolic and Algebraic Computation (ISSAC), July 19–22, 2016, Waterloo, Ontario, Canada.

1. INTRODUCTION

Computing the real roots of a univariate polynomial is one of the fundamental tasks in numerics and computer algebra, and numerous methods have been proposed to solve this problem. The leading general-purpose solvers in practice are based on subdivision algorithms that rely on Descartes' Rule of Signs to test for the existence of roots in a certain interval.

The computation for an input polynomial $P \in \mathbb{R}[x]$ can be considered as a binary tree where each node corresponds to an interval $I = (a, b)$ and a polynomial $P_I = (x+1)^n P(\frac{ax+b}{x+1})$. The number v_I of sign changes in the coefficient sequence of P_I then exceeds the number of roots contained in I by an even non-negative number. In the original algorithm [2], the two children of a node are obtained by a simple bisection on the interval and relative transformations on the polynomials, which yields the polynomial P_I. A node is a leaf if $v_I = 0$ or $v_I = 1$; in the first case, I contains no root, whereas, in the latter case, the interval is isolating for a real root. Considerable efforts have been taken to improve the worst-case complexity of different variants of the Descartes method, and to provide efficient implementations: different traversal orders of the subdivision tree to optimize the memory usage [10], the use of interval arithmetic [8], new strategies to optimize the main transformations [16], extensions to polynomials with approximate coefficients [11, 5], and many others.

Nevertheless, most formulations suffer from at least one of the following two deficiencies: First, the subdivision strategies only achieve *linear* convergence against the roots. In presence of clusters, those approaches require a large number of subdivisions to separate the roots. Second, the algorithms are designed for exact arithmetic on the coefficients of P. An unguarded choice of the subdivision points can impose an unnecessarily large precision demand when such an approach is translated literally to arbitrary precision dyadic numbers. Even worse, completeness is not guaranteed for polynomials whose coefficients can only be approximated. For example, consider the polynomial $P(x) = (x - 1)(\pi x - e)$ with roots at (exactly) 1 and $e/\pi \approx 0.865$, and assume that its coefficients are given as an oracle for arbitrarily good dyadic approximations. In this setting, $P(1) = 0$ cannot be decided without resorting to a symbolic simplification of the input, which is beyond the means of the root isolation method. Likewise, the number of sign variations on the intervals $(1 - \epsilon, 1)$ and $(1, 1 + \epsilon)$ for any ϵ cannot be determined with approximate arithmetic. Thus, once an interval is split at $x = 1$, a straight-forward algorithm will request better and better approximations of P from the oracle, but will not terminate.

Both shortcomings have been resolved in [19]: The proposed algorithm ANewDsc combines the bisection strategy with Newton iteration to achieve *quadratic* convergence against clusters. Thus, long chains of intervals I with the same v_I are compressed to only logarithmic length (compared to the length when considering bisection only). In addition, intervals are split at *admissible points,* where the polynomial takes a (relatively) large absolute value. This allows the precision demand of the computation to be kept small, improving by an order of magnitude over previous approaches. Due to the exclusive use of approximate arithmetic, the method also applies to (square-free) polynomials with arbitrary real coefficients. Both the number of subdivision steps and the precision demand is optimal up to polylogarithmic factors, and the bound on its bit complexity is comparable to the record bound [12] that is implied by an algorithm based on Pan's near-optimal method [14] for approximate polynomial factorization. We provide more specific bounds in Section 2.

Unfortunately, asymptotically fast algorithms are often extremely hard to be implemented or do not exhibit their theoretical performance in practice. The contribution of this paper is to show that, for the problem of real root computation, we can bridge this gap between theory and practice. We report on an implementation of ANewDsc on top of the Descartes-based real root finder inside the RS library. It preserves the key features of RS, which are a close-to-optimal memory consumption and the intensive use of adaptive multiprecision interval arithmetic [15].

We present our design changes both to RS and within ANewDsc that make it possible to achieve significant performance gains on hard instances on the one hand and proven complexity guarantees on the other hand without sacrificing efficiency for small or intrinsically easy instances. The analysis is supported by benchmark results that show that ANewDsc can defy the leading general-purpose solvers for real roots in their special domains, and outperforms the existing implementations on notoriously hard instances.

2. THE DESCARTES METHOD AND THE ANewDsc ALGORITHM

We briefly review the classical Descartes method [2] as well as the algorithm ANewDsc as introduced in [19]. Given an interval $\mathcal{I} = (a_0, b_0) \subset \mathbb{R}$ and a polynomial

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

the Descartes method recursively subdivides $\mathcal{I}$ into equally sized intervals until each subinterval $I = (a, b) \subset \mathcal{I}$ has either been shown to contain no root or exactly one root of P. In order to test an interval for the existence of a root of P, a coordinate transformation $x \mapsto \frac{ax+b}{x+1}$ is considered, which maps $\mathbb{R}^+$ one-to-one onto the interval I. Then, Descartes' Rule of Signs applied to the polynomial

$$P_I(x) := \sum\nolimits_{i=0}^{n} p_{I,i} x^i := (x+1)^n \cdot P\Big(\frac{ax+b}{x+1}\Big) \qquad (2)$$

states that the number of sign changes $v_I := \operatorname{var}(P, I) := \operatorname{var}(P_I)$ in the coefficient sequence of P_I exceeds the number m_I of roots (counted with multiplicity) of P in I by a non-negative even integer. In other words, $m_I \leq v_I$ and $m_I \equiv v_I$ mod 2. In each step of the recursion within the Descartes method, the number v_I is computed. If $v_I = 0$, the interval I is discarded. If $v_I = 1$, I is stored as isolating. Intervals with $v_I > 1$ are further subdivided into two equally sized subintervals. In addition, it is checked whether the subdivision point, that is the midpoint $m(I) = \frac{a+b}{2}$ of I, is a root of P.

Algorithm 1: Classical Descartes Method

```
INPUT: A polynomial $P$ as in (1) and an interval $\mathcal{I}$.
REPORTS: Disjoint isolating intervals for all roots of $P$ in $\mathcal{I}$.
• Initialize the list of active intervals to $\mathcal{A} := \{\mathcal{I}\}$.
• While $\mathcal{A} \neq \emptyset$:
  ▷ Remove an arbitrary $I = (a, b)$ from $\mathcal{A}$.
  ▷ If $v_I = 0$, discard $I$ and continue.
  ▷ If $v_I = 1$, report $I$ and continue.
  ▷ Otherwise, add $(a, m(I))$ and $(m(I), b)$ to $\mathcal{A}$.
        If $P(m(I)) = 0$, additionally report $\{m(I)\}$.
```

If the polynomial P contains only simple roots in $\mathcal{I}$, the Descartes method yields isolating intervals for all these roots; otherwise, it converges towards the roots, but does not terminate. If $\mathcal{I}$ is chosen large enough to contain all real roots, and all these roots are simple, the algorithm isolates all real roots of P. The proof for termination relies on the well known One- and Two-Circle Theorems [4, 13], which provide lower bounds on the width $w(I)$ of any produced interval I.

Regarding the worst-case complexity of the above algorithm, we focus, only for simplicity, on the so-called *benchmark problem* of computing all real roots of a square-free polynomial P of degree n with *integer* coefficients of absolute value less than 2^τ. Nevertheless, we aim to stress the fact that, for polynomials with arbitrary real coefficients, more general bounds are known [18, 19], which are expressed in terms of the separations and the absolute values of the roots of P, thus being more adaptive and meaningful.

We denote by $\mathcal{T}$ the subdivision tree whose nodes are the intervals I considered by the algorithm. The function $\operatorname{var}(\cdot)$ is sign diminishing, that is, for any disjoint subintervals I_1 and I_2 of I, we have $v_{I_1} + v_{I_2} \leq v_I$; e.g. see [4] for a proof. This implies that $\mathcal{T}$ has width at most $2n$, while the lower bound on the width of the produced intervals implies that its height is in $O(\tau + \log \sigma_P^{-1})$, where σ_P is the minimum root separation of P. Since $\log \sigma_P^{-1} = \tilde{O}(n\tau)$, the bound $\tilde{O}(n^2\tau)$ on the total size of $\mathcal{T}$ follows.[1] A more refined argument [6, 4], which takes into account the fact that $\sum_{i=1}^{n} \log \sigma_P(z_i)^{-1} = \tilde{O}(n\tau)$, even shows that $|\mathcal{T}| = \tilde{O}(n\tau)$. The coefficients of the polynomials P_I have bitsize bounded by $\tilde{O}(n^2\tau)$, hence the precision demand for exactly computing P_I is also bounded by $\tilde{O}(n^2\tau)$. This yields the bound $\tilde{O}(n^3\tau)$ for the bit complexity of computing P_I, even when using asymptotically fast methods for polynomial arithmetic. We conclude that the cost of the classical Descartes method is bounded by $\tilde{O}(n^4\tau^2)$ bit operations. This matches the bound achieved by many other popular subdivision algorithms for real root computation such as the continued fraction (CF) method [1, 20], the Bolzano method [21], or the Sturm method [3]. We remark that the $\tilde{O}(n^4\tau^2)$ upper bound on the bit complexity is actually tight for the latter algorithms; see [6, 1, 3].

The main strengths of the Descartes method are its simplicity and the fact that tree subdivision tree size and running time adapt to the separation of the roots [6, 4, 18]. On well-conditioned inputs, the performance is significantly better

[1]The tilde denotes that we omit polylogarithmic factors in n or τ.

than indicated by the worst-case bounds. Indeed, we observe that, for many polynomials (e.g. random polynomials), the size of the tree is only logarithmic in n, in which case the precision demand does not exceed $\tilde{O}(\tau + n)$. In such cases, the bit complexity is by several magnitudes smaller than the worst case bound. However, despite its good behavior for many instances, the method has two major shortcomings.

First, in order to determine the sign of the coefficients of P_I as well as the sign of P at the midpoint of I, exact arithmetic is assumed. If the coefficients of P are integer or rational, this is feasible, at the expensive of a precision that exceeds the actual demand by one order of magnitude; see Table 1 and [18, 19] for more details. However, if the coefficients of P can only be approximated (e.g trigonometric expressions), then computing the sign of the coefficients of P_I is impossible in general. Another shortcoming is that the classical subdivision strategies only achieve linear convergence against the roots. For some inputs, this is not critical as the separations of the roots are large and the subdivision tree has small height. However, if roots appear in clusters, numerous subdivision steps have to be performed in order to separate distinct roots from each other, and there exist polynomials[2] for which the Descartes method produces a sequence of intervals I_j of length $\Omega(n\tau)$ with identical v_{I_j}.

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

All mentioned bounds are tight for certain classes of inputs.
*Amortized precision demand over the entire tree is only $\tilde{O}(n\tau)$.
†Amortized precision demand over the entire tree is only $\tilde{O}(n+\tau)$.

Table 1: worst-case complexity of variants of Descartes methods for polynomials in $\mathbb{Z}[x]$ with degree n and bitsize τ

In [19], the algorithm ANewDsc[3] has been introduced. It uses approximate arithmetic and an accelerated subdivision strategy based on Newton's iteration to address both of the aforementioned shortcomings. We sketch the main ideas to an extent as needed in the discussion of its implementation. At some points, we simplified the presentation at the cost of mathematical rigor by skipping some rather technical details. In contrast, our implementation takes into account all details, thus being certified and complete.

ANewDsc is similar to the classical Descartes method in the sense that it recursively subdivides a given interval $\mathcal{I}$ and that it uses a predicate based on Descartes' Rule of Signs to test for roots. However, it is tailored to rely solely on approximate computation. For this, ANewDsc uses the so called "01-Test",[4] that can be evaluated using approximate arithmetic only. Similar to the original Descartes test, a call of the predicate on an interval I returns a value $t_I \in \{0, 1, *\}$

[2]As an example, consider the Mignotte-like polynomials from Table 2 having two real roots with pairwise distance $2^{-\Omega(n\tau)}$.

[3]The algorithm is an approximate arithmetic variant of the algorithm NewDsc from [17], which exclusively uses exact arithmetic. The acronym ANewDsc should be read as "A New Descartes" or, alternatively, as "Approximate Arithmetic Newton-Descartes."

[4]The 01-Test is split into two separate subroutines. Both are modified variants of the classical sign variation tests, where it is checked whether $\mathrm{var}(P, J) = 0$ or $\mathrm{var}(P, J) = 1$ for some intervals J. For details and proofs regarding the 01-Test, we refer the reader to [19, Sec. 2.4].

Algorithm 2: Find Admissible Point

```
INPUT: Polynomial $P$ as in (1) and a multipoint $m[\epsilon; N]$.
OUTPUT: Admissible point $m^* \in m[\epsilon; N]$.
• For $\rho = 2, 4, 8, \dots$:
  ▷ For all $m_i \in m[\epsilon; N]$, compute approximations $\tilde{v}_i$
    of $v_i := P(m_i)$ with $|\tilde{v}_i - v_i| < 2^{-\rho}$.
  ▷ Determine a point $m_{i_0} = m_i$ that maximizes $|\tilde{v}_i|$.
  ▷ If $|\tilde{v}_{i_0}| > 2^{-\rho+2}$, return $m^* := m_{i_0}$.
```

with the following guarantees: If $t_I = 0$, then I contains no root of P; if $t_I = 1$, then I isolates a simple root of P; and if $t_I = *$, no conclusion shall be drawn about the number of roots of P in I. We remark that the 01-Test is stronger than the classical sign variation test in the sense that $v_I = 0$ (or 1) implies that the 01-Test returns 0 (or 1), too.

The precision demand of the 01-Test on (a, b) is bounded by $\tilde{O}(n + n\log|a| + n\log|b| + \log|P(a)|^{-1} + \log|P(b)|^{-1} + \log\|P\|_\infty)$ bits.[5] Hence, we strive to choose boundary points a and b where the value of $|P|$ is not too small. This is realized by an additional layer on the subdivision scheme. Instead of choosing a fixed subdivision point m (in the classical method, the midpoint $m(I)$ of an interval I) in each step, ANewDsc chooses a nearby, so-called *admissible point* m^*, where $|P|$ becomes large. More precisely, for a point m, a positive integer N with $N \geq n$, and a positive value ϵ, we call a point

$$m^* \in m[\epsilon; N] := \{m_i := m + i \cdot \epsilon \text{ for } i = -\lceil \tfrac{N}{2} \rceil, \dots, \lceil \tfrac{N}{2} \rceil\}$$

admissible with respect to the multipoint $m[\epsilon; N]$ *(or just admissible)* if $|P(m^*)| \geq \frac{1}{4} \cdot \max_i |P(m_i)|$. Algorithm 2 computes such a point using only approximate arithmetic.[6] Indeed, since $m[\epsilon; N]$ contains at least one point that has distance $\epsilon/2$ or more to all roots of P, we have $\max_i |P(m_i)| > 0$, and thus the algorithm terminates with a precision ρ that is bounded by $2 \cdot (1 + \log\max(1, (\max_i |P(m_i)|)^{-1}))$.

In [19], we always choose $N = n$. Using approximate multipoint evaluation [9], this guarantees that the cost for the approximate evaluations of P at all points m_i does not considerably exceed the cost for the evaluation at only one point. In addition, in each step, ϵ is chosen small enough (e.g., $\epsilon \approx w(I)/4$ in a bisection step with $m = m(I)$) such that the size and the structure of the subdivision tree induced by ANewDsc does not change (in the worst case) when passing from an admissible point $m^* \in m[\epsilon; N]$ to an arbitrary point in the interval $[m - \lceil \frac{N}{2} \rceil\epsilon, m + \lceil \frac{N}{2} \rceil\epsilon]$. In fact, choosing m^* to be admissible only affects the needed precision demand in each iteration but not the size of the subdivision tree.

For simplicity, we will not further specify ϵ (nor N) throughout the following considerations and just assume that ϵ is chosen small enough; for details, see [19]. In this context, we just say that a point m^* is *admissible for* m. With a suitable choice of the parameters, subdivision on admissible points reduces the worst-case precision demand of the Descartes method to $\tilde{O}(n\tau)$ in the integer setting, improving upon the classical approach by one order of magnitude. Besides, it allows to process inputs where the coefficients can only be approximated as we avoid splitting on roots *without* verifying that P is zero at any point.

[5]We remark that the precise bound on the precision demand is slightly more complicated; see [19, Sec. 2.4] for details.

[6]Note that in [19, p. 54], there is a typo in the Admissible Point algorithm: *repeat ... until* (loop (2)) should read *repeat ... while*. However, the proof of [19, Thm. 5 (b)] uses the correct statement.

In order to overcome the second shortcoming of the Descartes method, ANewDsc combines bisection with Newton iteration. For this, it uses a trial and error approach, called "Newton-Test", to speed up convergence towards clusters of roots. In each iteration, the Newton-Test aims to replace an interval $I = (a, b)$ by some sub-interval $I' = (a', b') \subset I$ of width $w(I') \approx w(I)/N_I$ such that I' contains all roots of P in I. Here, N_I denotes a parameter that corresponds to the actual speed of convergence. Initially, N_I is set to 4. If the Newton step succeeds, we define $N_{I'} := N_I^2$. In case of failure, we fall back to bisection, that is, I is subdivided into two (almost) equally-sized subintervals I_ℓ and I_r, and we define $N_{I_\ell} := N_{I_r} := \max(4, \sqrt{N_I})$. In [19, Sec. 3.2], it has been shown that the Newton step succeeds under guarantee if there exists a sufficiently small cluster $\mathcal{C}$ of roots of P that is centered at some point in I and is sufficiently well-separated from the remaining roots of P.

The method runs in three steps. In the first step, it estimates the multiplicity k of such a cluster.[7] Then, in the second step, it performs a corresponding Newton iteration to obtain a λ with $|\lambda - z_i| < w(I)/2N_I$ for each root $z_i \in \mathcal{C}$. Finally, the method aims to validate the approximation λ of the cluster $\mathcal{C}$. For this, we apply the 01-Test to the intervals (a, a') and (b, b'), where $I' = (a', b') \subset I$ is an interval of width $w(I') \approx w(I)/N_I$ centered at λ. If the latter test yields the value 0 for both intervals (a, a') and (b, b'), it follows that I' contains all roots that are contained in I, in which case we say that the Newton-Test succeeds. Notice that, even in case of success, we may not conclude that there exists a cluster of k roots, however, we may conclude that I may be replaced by I' without discarding any root.

It should further be mentioned that the Newton-Test also integrates a so-called Boundary-Test, which, by default, always checks whether one of the two intervals $(a, a + w(I)/N_I)$ or $(b - w(I)/N_I, b)$ contains all roots that are contained in I. Again, for this, the 01-Test is applied to $(a + w(I)/N_I, b)$ and $(a, b - w(I)/N_I)$, respectively; see Algorithm 4 and [19, Sec. 3.2] for more details.

In [19], the bit complexity bit complexity as well as the bounds on the number of iterations and the precision demand of ANewDsc are stated in terms of the degree of P and values that exclusively depend on the roots z_i of the polynomial, such as the product of the absolute values of all roots beyond the unit disk or the product of the pairwise distances between any two roots. Those bounds are both more adaptive to the intrinsic complexity of the problem and apply for inputs with approximable coefficients. For brevity, we only restrict to the benchmark problem here: The length of a sequence of intervals I_j with invariant $\operatorname{var}(P, I_j)$ is now upper bounded by $\tilde{O}(\log(n\tau))$ compared to $\tilde{O}(n\tau)$ for the classical Descartes method. This is due to the fact that the Newton-Test succeeds and achieves quadratic convergence for all but $\tilde{O}(\log(n\tau))$ intervals. As a consequence, the overall size of the subdivision tree is bounded by $O(v_{\mathcal{I}} \cdot \log(n\tau)) = O(n \log(n\tau))$, which is near-optimal; see also [19, Theorems 27–29]. The bit complexity of the algorithm is bounded by $\tilde{O}(n^3 + n^2\tau)$, which is by three magnitudes better than the bound for the Descartes method (also see Table 1) and comparable to the best bound known [12, 7] for the benchmark problem as achieved by an algorithm based on Pan's near-optimal method [14] for approximate polynomial factorization. The bound is obtained by an amortized analysis of the cost at each node in the subdivision tree. The cost for processing a certain interval I is related to the maximum of all products $\max_{z_i} \sigma_P(z_i) \cdot |P'(z_i)|$, where the z_i's range over all roots in the one-circle region of I. In the worst-case, the cost at a node is of size $\tilde{O}(n^2\tau)$, whereas, in average, it is of size $\tilde{O}(n(n + \tau))$. The precision demand is upper bounded by $\tilde{O}(n\tau)$ in the worst case and of size $\tilde{O}(n + \tau)$ in average.

Algorithm 3: Newton-Test

```
INPUT: A polynomial $P$ as in (1), an interval $I = (a, b)$, and
       an $N_I \in \mathbb{N}$ of the form $N_I = 2^{2^l}$ with $l \in \mathbb{N}_{\geq 1}$.
OUTPUT: FALSE or an interval $I' \subset I$, with $\frac{N_I w(I')}{w(I)} \in [\frac{1}{8}, 1]$,
        that contains all roots of $P$ in $I$.
• For $j \in \{1, 2, 3\}$, let $\xi_j := a + \frac{j}{4} \cdot w(I)$,
  compute admissible points $\xi_j^*$ for $\xi_j$ and
  the Newton correction terms $v_j := P(\xi_j^*)/P'(\xi_j^*)$.
• For all pairs $i, j \in \{1, 2, 3\}$ with $i < j$:
  ▷ Compute approximations $\tilde{\lambda}_{i,j}$ of
        $\lambda_{i,j} := \xi_i^* + \tilde{k} \cdot v_i$, where $\tilde{k} := (\xi_j^* - \xi_i^*)/(v_i - v_j)$,
    with $|\tilde{\lambda}_{i,j} - \lambda_{i,j}| \leq 1/32N_I$.
  ▷ If $\tilde{\lambda}_{i,j} \notin [a, b]$, discard $(i, j)$. Otherwise, define
        $\ell_{i,j} := \lfloor (\tilde{\lambda}_{i,j} - a) \cdot 4N_I/w(I) \rfloor \in \{0, \ldots, 4N_I\}$,
        $a_{i,j} := a + \max\{0, \ell_{i,j} - 1\} \cdot w(I)/4N_I$, and
        $b_{i,j} := a + \min\{4N_I, \ell_{i,j} + 2\} \cdot w(I)/4N_I$.
    If $a_{i,j} = a$, set $a_{i,j}^* := a$, and if $b_{i,j} = b$, set $b_{i,j}^* := b$.
    Otherwise, compute admissible points $a_{i,j}^*$ and $b_{i,j}^*$
    for $a_{i,j}$ and $b_{i,j}$.
  ▷ If the 01-Test returns 0 for both intervals $(a, a_{i,j}^*)$
    and $(b_{i,j}^*, b)$, return $I' = (a_{i,j}^*, b_{i,j}^*)$.
    Otherwise, discard the pair $(i, j)$.
• Boundary test: If all pairs have been discarded,
  compute admissible points $m_\ell^*$ and $m_r^*$ for
  $m_\ell := a + w(I)/2N_I$ and $m_r := b - w(I)/2N_I$.
  ▷ If the 01-Test yields 0 for $(m_\ell^*, b)$, return $(a, m_\ell^*)$.
  ▷ If the 01-Test yields 0 for $(a, m_r^*)$, return $(m_r^*, b)$.
  ▷ Otherwise, return FALSE.
```

This is a simplified version of Newton- and Boundary-Test from [19]. For the sake of a concise presentation, some technical details (e.g. concerning the precision management) are omitted.

3. IMPLEMENTING ANEWDSC INSIDE RS

The RS library contains a generic framework [16] that allows to instantiate several variants of the Descartes method. The choices include several bisection variants as well as continued fraction-based subdivision. The default configuration, which we will denote as "RS" throughout the following considerations, achieves its high efficiency by a careful low-level implementation of the bottleneck subroutines as well as a few sophisticated and—at that time—innovative design choices.

RS uses a hybrid arithmetic strategy. Starting with a low precision (of 63 bits), all predicates are evaluated in interval arithmetic using MPFI [15]. Whenever a predicate cannot be conclusively decided within the working precision, the computation is interrupted and later resumed with a higher precision. When it is conceivable that exact arithmetic is less costly than interval arithmetic at high precision, a heuristic can trigger exact computation; this facilitates further optimizations like (virtual) deflation of exact dyadic roots.

[7]Algorithm 3 only computes some rational value $\tilde{k}$. Under the assumption that a cluster $\mathcal{C}$ as above exists, we have $k \approx \tilde{k}$ and, in particular, k equals the integer that is closest to $\tilde{k}$.

Algorithm 4: ANewDsc

```
INPUT: A polynomial $P$ as in (1) and an interval $\mathcal{I}$.
REPORTS: Disjoint isolating intervals for all roots of $P$ in $\mathcal{I}$.
• Initialize the list of active intervals to $\mathcal{A} := \{(\mathcal{I}, 4)\}$.
• While $\mathcal{A} \neq \emptyset$:
  ▷ Remove an arbitrary $(I, N_I)$ with $I = (a, b)$ from $\mathcal{A}$.
  ▷ If the 01-Test on $I$ returns 0, discard $I$ and continue.
  ▷ If the 01-Test on $I$ returns 1, report $I$ and continue.
  ▷ Otherwise:  Quadratic (or Newton) step
    If the Newton-Test on $P$ and $(I, N_I)$ returns
    an interval $I'$, add $(I', N_I^2)$ to $\mathcal{A}$ and continue.
  ▷ Otherwise:  Linear (or bisection) step
    Compute an admissible point $m^*$ for $m(I)$ and
    add $(I', N_{I'})$ and $(I'', N_{I''})$ to $\mathcal{A}$, where $I' = (a, m^*)$,
    $I'' = (m^*, b)$, and $N_{I'} := N_{I''} := \max(4, \sqrt{N_I})$.
```

The realization of the sign variation test in RS is adapted to this interval setting: While we work with exact polynomials $f = \sum f_i x^i$ in theory, the computations are executed on coefficient-wise interval approximations $[f] = \sum [f]_i x^i$ of f such that $f_i \in [f]_i$ for all i. In particular, the value of $\mathrm{var}(f)$ is replaced by a (super-)set $\mathrm{var}([f]) = \{v^-, \dots, v^+\}$ of possible sign variations for the family of polynomials in $[f]$, that is, $\mathrm{var}(g) \in \mathrm{var}([f])$ for all $g \in [f]$. In this model, a sign variation test for a polynomial $[f]$ succeeds if and only if $\mathrm{var}([f]) = \{0\}$, $\mathrm{var}([f]) = \{1\}$, or $0, 1 \notin \mathrm{var}([f])$. Otherwise, the accuracy is not sufficient to decide the branching strategy for the current interval, and the precision is increased. We keep this classical test as a filter in ANewDsc; if it does not succeed, we call the full 01-Test, which again uses the interval sign variation computation as a primitive.

Another crucial difference between RS and the previous variants of the Descartes method is the traversal of the subdivision tree. In Collins' and Akritas' [2] formulation, the subdivision tree is explored in a depth-first search strategy. This traversal allows to obtain some of the intermediate polynomials with comparatively little computational effort from previous results, but comes at the expense of storing a potentially very long list of active nodes. On the other hand, Krandick's variant [8] with breadth-first traversal is more memory-efficient, but requires more expensive arithmetic operations; see [16, Sec. 3]. The trade-off which proved optimal for RS was a depth-first search traversal in a near-constant-memory variant, oblivious of intermediate results. To keep the arithmetic cost of the algorithm close to the optimum for bisection methods, RS uses specifically tailored subroutines for reconstructing the discarded intermediate polynomials. They are supplemented by a custom garbage collector that ensures that only an insignificant amount of time is spent for memory management. For details of those design choices, we refer to the original description of RS [16, Sec. 4].

RS is a promising basis for an implementation of ANewDsc due to its high performance as a general-purpose solver and the availability in Maple. Also, the extensive use of interval arithmetic offers the flexibility to design a numerical root solver for polynomials with irrational coefficients that can be approximated arbitrarily well. However, to combine the theoretical improvements of the new algorithm with the long-evolved insights for a practical realization, modifications are required on both ends. In this section, we will describe the most crucial design decisions in this light.

3.1 Changes over the theoretical ANewDsc

Random sampling of pseudo-admissible points.

A tight analysis of the theoretical worst-case bit complexity of any variant of the Descartes method heavily relies on asymptotically fast algorithms for polynomial arithmetic. However, in practice, many asymptotically fast methods are only effective for inputs larger than a threshold that is well beyond what can realistically be handled with state-of-the-art solvers. Until now, careful implementations of the naive algorithms are more efficient except for artificial counterexamples.

The most prominent places where such tools are used in ANewDsc are the basis transformations from (1) to (2) (so-called *Taylor shifts*) and the approximate multipoint evaluations within the admissible point selections. With the advanced methods, both subroutines require only a near-linear number of arithmetic operations in the coefficient ring, compared to quadratically many for naive approaches. The Taylor shifts are intrinsic to any Descartes method (although we will discuss a partial remedy for some situations at the end of Section 3.2). But the multipoint evaluations in the "Admissible Point" routine can be avoided: In our initial description of ANewDsc, we choose an admissible point $m^* \in \mathbf{m} := m[\epsilon; n]$ such that $|P(m^*)| \geq \frac{1}{4} \cdot \max_{m_i \in \mathbf{m}} |P(m_i)|$. For our implementation, we propose to consider the multipoint $\mathbf{M} := m[\frac{\epsilon}{2^\lambda}; n \cdot 2^\lambda]$ instead, which spans the same range as $\mathbf{m}$ (i.e. the extremal points in $\mathbf{m}$ and $\mathbf{M}$ define the same interval $[m - \lceil \frac{n}{2} \rceil \epsilon, m + \lceil \frac{n}{2} \rceil \epsilon]$) but contains 2^λ times as many points as $\mathbf{m}$. Here, λ is a positive integer of size $O(\log n)$. However, instead of choosing an admissible point in $\mathbf{M}$ we choose a so-called *pseudo-admissible point*, where P is only required to take an absolute value that is related to the current working precision. For this, we randomly sample a point m_j from $\mathbf{M}$ and compute an approximation $\tilde{v}_j$ of $v_j = P(m_j)$ (via interval arithmetic) with $|\tilde{v}_j - v_j| < 2^{-\rho}$. If $|\tilde{v}_j| > 2^{-\rho+2}$, we keep m_j. Otherwise, we double the precision ρ and choose a new sample point; see also Algorithm 5.

Algorithm 5: Find Pseudo-Admissible Point

```
INPUT: A polynomial $P(x)$ as in (1) and a multipoint
       $\mathbf{M} := m[\frac{\epsilon}{2^\lambda}; 2^\lambda n]$ with $\lambda \in \mathbb{N}_{\geq 2}$ and $\lambda = O(\log n)$.
OUTPUT: A point $m' \in \mathbf{M}$ with value $P(m') \neq 0$.
• For $\rho = 63, 127, 255, \dots$ :
  ▷ Pick a random $m_j$ among the points in $\mathbf{M}$.
  ▷ Compute an approximation $\tilde{v}_j$ of $v_j = P(m_j)$
    with $|\tilde{v}_j - v_j| < 2^{-\rho}$ (using interval arithmetic).
  ▷ If $|\tilde{v}'| > 2^{-\rho+2}$, return $m' := m_j$.
```

The following lemma guarantees that, with high probability, we choose a point $m' \in \mathbf{M}$ for which $|P(m')|$ is not much smaller than $\max_{m_i \in \mathbf{m}} |P(m_i)|$.

Lemma 1. *Algorithm 5 returns a point $m' \in \mathbf{M}$ such that*

$$\log |P(m')| = 2^k \cdot O(n \log n + \log \max(1, (\max_{m_i \in \mathbf{m}} |P(m_i)|)^{-1}))$$

with probability $1 - 2^{-k\lambda} \geq 1 - 2^{-k}$.

Proof. Let $\mathbf{M}^* \subset \mathbf{M}$ denote the subset consisting of all points in $\mathbf{M}$ whose distance to all roots of P is at least $\epsilon \cdot 2^{-\lambda}$. Let $\bar{m} \in \mathbf{m}$ be a point with $|P(\bar{m})| = \max_{m_i \in \mathbf{m}} |P(m_i)|$, and define $\rho_0 := \lceil \log \max(1, |P(\bar{m})|^{-1}) \rceil$. For any complex root z of P and any point $m_j \in \mathbf{M}^*$, we have $\frac{|\bar{m} - z|}{|m_j - z|} > \frac{1}{1 + n 2^\lambda} >$

$2^{-\lambda-1-\log n}$ and, thus, $\frac{|P(m_j)|}{|P(\tilde{m})|} > 2^{-n(\lambda+1+\log n)}$. Suppose that, in a certain iteration of Algorithm 5, we have $\rho \geq \rho_0 + n(\lambda+1+\log n)+2$. Then, it holds that $2^{-\rho+2} < |P(m_j)|$ for all $m_j \in \mathbf{M}^*$; hence, the algorithm terminates if we pick such an m_j. Since $\mathbf{M}^*$ contains at least $2^\lambda(n+1)-2n$ points, the probability of choosing a point from $\mathbf{M}^*$ is at least $1-2^{-(\lambda-1)}$. Hence, with probability $1-2^{-k(\lambda-1)}$, we terminate with a ρ of size $2^k \cdot O(n\log n + \log\max(1, (\max_{m_i\in\mathbf{m}}|P(m_i)|)^{-1}))$. □

Our analysis shows that, in expectation, there might be an increase in precision by an additive term of size $O(n\log n)$ when choosing a pseudo-admissible point instead of an admissible point; in practice, we observe that the increase in precision is entirely negligible. We further remark that the complexity bounds as derived in [19] are not affected, since a term of size $\tilde{O}(n)$ already appears in the bound on the precision demand of the 01-Test. Yet, our implementation only models the theoretical result in a Las-Vegas setting due to the random choice of a pseudo-admissible point.

Heuristics to delay invocations of the Newton step.

The asymptotic cost of the calls to the Newton-Test is dominated by the Taylor shift which is part of the sign-variation test for any interval. However, in practice there is a noticeable impact of unsuccessful Newton-Tests on the performance due to the expensive 01-Tests for the siblings $(a, a^*_{i,j})$ and $(b^*_{i,j}, b)$ of the candidate intervals $(a^*_{i,j}, b^*_{i,j})$. Thus, we strive for calling the Newton-Test only if it will succeed with high likelihood; vice versa, in the generic situation of well-distributed roots without any distinguished root clusters, as little computation time as possible should be wasted.

We say that a linear step on an interval I to I' and I'' leads to a *proper split* if $\operatorname{var}(P, I')$ and $\operatorname{var}(P, I'')$ are both non-zero. In this case, Obreshkoff's One-circle theorem [4, 13] asserts that there is at least one root in the neighborhood of I'' that is not part of a potential cluster within I', and the symmetric argument holds for I''. There are two explanations for such a situation: If the goal on the convergence speed was too optimistic (that is, the value of N_I was set too high), the candidate interval in the Newton-Test might not comprise the entire cluster of roots. In this case, a success of the Newton step on I' and I'' with a coarser resolution $N_{I'} = N_{I''} = \sqrt{N_I}$ is not entirely unlikely. However, if $N_I = 4$ was already at the minimum, the roots near I' and I'' are well-separated compared to the resolution N_I, and an immediate success of the Newton-Test would be an unexpected artifact.

We installed a heuristic to distinguish those situations and inhibit Newton-Tests in the latter case. For each node I in the subdivision tree, we keep track of the distance to its closest ancestor J with $N_J = 4$ that lead to a proper split. If this distance is smaller than $\log n$, we immediately process I with bisection[8] and keep $N_I = 4$. On intervals with well-separated roots where we never achieve nor profit from quadratic convergence, this strategy renders it likely that there is never even an attempt to call a Newton-Test, and the overhead of ANewDsc compared to the variant ADsc without the Newton-Test is almost zero. On the other hand, the size of the subdivision tree is increased by at most a factor of size $O(\log n)$, and thus stays soft-linear in n and polylogarithmic in τ in the worst case.

[8]In the terminology of [19], *proper splits* occur on a subset of the *special nodes* of the subdivision tree, and we allow paths of $\lceil \log n \rceil$ many *ordinary nodes* before further Newton-Tests.

3.2 Changes over the classical RS

Full caching instead of constant-memory version.

The default instantiations of RS use a near-constant-memory strategy. None of the intermediate local polynomials are cached, but are computed from the previously considered node. In general, this means that larger round-off errors are accumulated, and expensive recomputation from scratch is potentially required more often. Yet, performance comparisons show a clear benefit of this approach due to the drastically reduced costs for memory management and RS' sophisticated way of performing incremental computations between adjacent nodes in the subdivision tree [16, Sec. 4].

However, the fast transformations rely on certain structural properties of the subdivision tree: in a pure bisection algorithm, all emerging intervals are of the form $(\frac{c}{2^e}, \frac{c+1}{2^e})$ for some integers c and e. In this setting and with the strong specification of the traversal order of the tree, a good portion of the arithmetic operations for Taylor shifts can be achieved by cheap bitshift and addition operations. In contrast, a general Taylor shift requires noticeably more expensive multiplications of arbitrary precision interval values.

These relations between the intervals are lost when Newton steps are performed or the canonical dyadic subdivision points need to be replaced by (pseudo-)admissible points. Furthermore, optimizations such as the delayed Newton calls are incompatible with strategies that keep only one interval in memory at a time, because the behavior of the algorithm on some interval is no longer independent of the neighborhood: whether a quadratic convergence step is pursued becomes conditional on the outcome of the 01-Test on its sibling.

Fortunately, using the Newton technique, long chains of intervals with a unique descendant in the bisection tree are compressed to only logarithmic height, thus shrinking the entire tree by an order of magnitude in the worst-case and even two orders of magnitude for some special instances (e.g. sparse polynomials with clustered roots such as Mignotte polynomials). In this light, the impact of the constant-memory strategy is much less pronounced, and it is advisable to switch back to the naive approach of caching all intermediate results. We stress that, without the Newton iteration, the memory consumption in a depth-first traversal would be prohibitive for particular examples with strongly clustered, ill-separated roots such as Mignotte polynomials.

Degree truncation through partial Taylor shifts.

We can consider any univariate polynomial $P(x)$ as a function $P(x) = p_n \cdot \prod_{j=1}^n (x - z_j)$ in its (not necessarily distinct) complex roots z_j. When the domain of interest is restricted to a small region in the complex plane, say, a disk $D \subset \mathbb{C}$, the relative influence of each root z_j on P scales with the distance of z_j to the points $x \in D$. If there are only $k \ll n$ roots in D and all other roots are well-separated from D, the distance to the remote roots is almost stable for all $x \in D$, and $P|_D$ is dominated by the roots in D.

In this situation, the local behavior of P is captured in its partial Taylor expansion around a point within D up to the k-th term. Hence, an approximation of P by its truncated Taylor expansion might suffice for computing isolating intervals for the real roots. We consider this approach whenever a Newton step succeeds and suggests the existence of a well-separated cluster of $k \ll n$ roots around an interval I. We remark that the multiplicity guess $k := \operatorname{round}(\tilde{k})$ is already computed within the Newton-Test. To ensure correctness, we conservatively take into account the truncation error as the interval evaluation of the Lagrangian remainder term.

More precisely, whenever the inclusion-exclusion-predicates of ANewDsc call for sign variation tests on some P_I, we work on an approximation for the intermediate polynomial $Q_I := \sum_{i=0}^{n} q_{I,i}x^i := P(a + (b-a)x)$ on $I = (a, b)$,

$$\tilde{Q}_I(x) = \sum\nolimits_{i=0}^{k-1} q_{I,i}x^i + [r]_{I,k}x^k, \quad [r]_{I,k} \supset \frac{P^{(k)}(I)}{k!},$$

where the coefficient $[r]_{I,k}$ of the Lagrangian remainder is evaluated on the entire interval I in conservative interval arithmetic. The root exclusion test on the truncated polynomial remains almost identical; however, we modify the certificate for inclusion of a single root.

Lemma 2. *Let P be a polynomial as in (1) and $I = (a, b)$ and $\tilde{Q}_I$ be as above. Define $\tilde{G}_I := (x+1)^k\ \tilde{Q}_I((x+1)^{-1})$ and $\tilde{H}_I := (x+1)^{k-1}\ \tilde{Q}'_I((x+1)^{-1})$.*

1. *If $\tilde{v}_I := \text{var}(\tilde{G}_I) = \{0\}$, then P has no root in I.*[9]
2. *If $\tilde{v}_I = \{1\}$ and, additionally, $\tilde{v}'_I := \text{var}(\tilde{H}_I) = \{0\}$, then P has exactly one simple real root in I.*

Proof. We consider the polynomials $\tilde{Q}_I$, $\tilde{G}_I$ and $\tilde{H}_I$ simultaneously as polynomials with interval coefficients and as the set of exact polynomials with coefficients contained in the intervals. Assume that $\tilde{v}_I = 0$, but P has a root $\xi = a + \lambda(b-a) \in I$. By the Lagrange representation of the truncation error in Taylor's theorem, there exists a $\xi \in I$ such that the polynomial $F_I(x) := \sum_{i=0}^{k-1} q_{I,i}x^i + \frac{P^{(k)}(\xi)}{k!}x^k$ of degree k has a root at λ. Hence, Descartes' Rule of Signs asserts $\text{var}((x+1)^k\,F_I((x+1)^{-1})) > 0$. Since $F_I \in \tilde{Q}_I$, it follows that $(x+1)^k\,F_I((x+1)^{-1}) \in \tilde{G}_I$ because the transformations in interval arithmetic overestimate the possible values. This contradicts the assumption that $\tilde{v}_I = 0$.

For the second part, an analogous argument on the derivative shows that P is strictly increasing or strictly decreasing on I if $\tilde{v}' = \{0\}$. Since $\tilde{v}_I = \{1\}$, we know that P takes values of different sign at the endpoints of I; hence, P has exactly one root in I. $\square$

If a 01-Test in the processing of a node is unsuccessful, we usually double the working precision. However, if the local polynomials are only partial Taylor expansions, the source of the loss in accuracy may also stem from the truncation. Hence, before increasing the working precision, we gradually double the multiplicity guess k until we eventually compute the full polynomial. This approach guarantees that, in the worst case, after $\log n$ steps the heuristic falls back to the theoretical model. Hence, even if the heuristic is unsuccessful, the complexity increases at most by a factor of $\log n$.

We observe that partial Taylor shifts give a tremendous speedup for polynomials with tight clusters of low multiplicity. In such situations, we can consider the technique as a partial substitute for the current deficiency of asymptotically fast Taylor shifts. For instances with clusters consisting of constantly many roots, the expected speedup is linear in n: if the heuristic applies, the local polynomials on each active region in the subdivision tree have to computed up to only constantly many coefficients of the Taylor series. This prediction is accurately reflected in the benchmarks.

[9]Recall that var(·), called on an interval polynomial, returns a (super-)set of the possible numbers of sign variations; see the remarks about the use of interval arithmetic in RS in Section 3.

4. EXPERIMENTS

We present and discuss a short excerpt of our extensive benchmark suite; the entire collection is available online at http://anewdsc.mpi-inf.mpg.de/. The objective is to illustrate that the integration of ANewDsc into RS entails very small overheads for instances with well-distributed roots and shallow subdivision trees, but huge performance gains in challenging situations with clustered roots.

The classical RS serves as a baseline in a configuration similar to the one used in Maple, but without some crucial optimizations to improve comparability. In particular, we disabled the hardware floating-point phase, which is not yet supported for ANewDsc, as well as the heuristic for switching to exact arithmetic that can improve performance, but destroys the guarantees on the expected precision demand. Besides the full-fledged ANewDsc (denoted as AND in the tables), we also list the intermediate variant ADsc. It includes subdivision on pseudo-admissible points and full caching, but neither Newton iterations nor degree truncation.

We compare to three well-established competitors:

MPSolve (named MPS in the tables) is the leading complex root solver. It is known to keep up with the most efficient real root isolators even though it solves a more general problem. Given our more modest goals, we restrict the region of interest to the real line, and call MPSolve in its latest version 3.1.5 with `unisolve -au -Gi -SR -Dr -Of -j1 -o1048576`.

CF denotes the *continued fraction*-based variant of the Descartes method available in *Mathematica* 10. We bypass the high-level interface of the computer algebra system via `System‘Private‘RealRoots` to eliminate the effect of preprocessing stages that are applied by the usual `RootIntervals` call (e.g. the detection and special handling of sparse polynomials or simplifications for even or odd polynomials).

Finally, we compare against Carl Witty's variant of Eigenwillig's bitstream Descartes method in Bernstein basis [5], which constitutes the default real root isolator in the *Sage* 7.0 open-source computer algebra system. We invoke the algorithm in the optimized variant for 64-bit architectures with `real_roots (f, skip_squarefree=True, wordsize=64)`.

All instances are passed as dense integer polynomials, filtered in a preprocessing stage to factor out side effects of optimizations unrelated to the solver. It involves the trivial algebraic simplification of even and odd polynomials, verification of square-freeness and reduction to the primitive part. The values in the tables are runtimes in seconds, measured on a single run on the same machine. Degree and coefficient bitsize are denoted by n and τ.

Polynomials of Mignotte type are classical benchmark instances. Their ill-separated pair of roots at approximately $2^{-\tau/2} \pm 2^{-n\tau/2}$ forces a huge subdivision depth in bisection approaches. This behavior is mitigated by the quadratic convergence in ANewDsc: The subdivision tree size shrinks from approximately 33500 nodes for RS and ADsc to a mere 47 for ANewDsc for $(n, \tau) = (129, 512)$, and even an instance with $\tau = 2^{16}$ leads to a tree with only 65 nodes. For such instances, CF presumably exhibits an almost optimal subdivision tree due to the rational center of the cluster; however, the performance is spoiled by the use of exact arithmetic. The class of nested Mignotte polynomials, shown around an irrational center in Table 3, shows the robustness of ANewDsc both to the higher multiplicity 20 and the irrational center of the cluster.

n	τ	MPS	CF	Sage	RS	ADsc	AND
257		0.7	0.1	1.6	7.6	7.7	0.1
513		2.9	0.2	2.4	87.6	89.1	0.2
1025		13.8	1.1	4.8	>*600*	>*600*	0.7
2049	14	78.5	7.2	12.8			2.9
4097		486.3	67.6	85.8			11.2
8193		>*600*	224.3	>*600*			43.2
16385			>*600*				188.1
	128	1.2	0.3	0.5	26.2	25.1	0.1
	512	5.9	3.8	3.9	378.5	293.8	0.4
	2048	43.4	78.4	>*600*	>*600*	>*600*	3.3
129	8192	274.7	>*600*				20.8
	32768	>*600*					96.8
	131072						503.0

Table 2: Mignotte: $x^n - ((2^{\tau/2} - 1)x - 1)^2$

n	τ	MPS	CF	Sage	RS	ADsc	AND
260		2.7	1.0	1.9	1.6	1.7	0.4
516		9.5	21.6	3.3	18.0	18.0	0.8
1028	140	67.8	565.0	>*600*	230.3	232.4	7.1
2052		283.6	>*600*		>*600*	>*600*	13.1
4100		>*600*					88.4
8196							394.0
	160	3.1	1.4	0.3	2.2	2.3	0.9
	640	5.8	14.9	*call*	20.8	19.2	0.9
260	2560	33.7	222.6	*stack*	301.2	254.0	3.7
	10240	248.0	>*600*	*over-*	>*600*	>*600*	31.4
	40960	>*600*		*flow*			341.2

Table 3: nested Mignotte: $\prod_{i=1}^{4} \left(x^{n/4} - ((2^{\tau/8} - 1)x^2 - 1)^{2i}\right)$

n	τ	MPS	CF	Sage	RS	ADsc	AND
1024		2.5	0.3	4.1	0.6	0.6	0.5
2048		9.8	0.9	8.4	1.6	1.7	1.7
4096	1024	37.5	2.7	29.1	3.8	3.7	3.5
8192		159.4	22.1	183.5	36.0	36.3	36.5
16384		578.9	376.6	>*600*	275.1	280.9	279.0

Table 4: dense with uniformly random coefficients in $(-2^\tau, 2^\tau) \cap \mathbb{Z}$

n	τ	MPS	CF	Sage	RS	ADsc	AND
256	506	4.7	34.1	1.6	18.2	19.1	1.1
512	762	25.9	456.9	3.7	107.7	110.4	3.0
1024	1274	207.5	>*600*	22.5	>*600*	>*600*	23.6
2048	2296	>*600*		>*600*			132.2
4096	4343						596.6

Table 5: clustered: $f^2 - 1$ for $f = \sum_i a_i \binom{n}{i}^{1/2} x^i / \sqrt{i+1}$ with a_i drawn from a normal Gaussian distribution, rounded to $\mathbb{Z}[x]$

Random polynomials have a low number ($\Theta(\log n)$) of well-separated real roots; they require only low precision for the isolation and induce flat subdivision trees. Hence, we can expect no gain from the improvements in ANewDsc. On the other hand, the experiments confirm that the enhancements incur almost no additional cost. We observe similar overheads of low constant factors compared to RS on other classes of polynomials with well-distributed roots, such as Wilkinson polynomials with evenly spaced real roots, different classes of orthogonal polynomials, or resultants of random bivariate polynomials that arise in projection-based polynomial system solving and the topology analysis of real algebraic curves.

Finally, polynomials with normal Gaussian-distributed coefficients have a much higher number ($\Theta(\sqrt{n})$) of real roots. By squaring and perturbing such inputs, we construct benchmark instances with many clusters of multiplicity two. We find that ANewDsc quickly detects the clusters, succeeds in the Newton steps, and computes the appropriate number of terms in the partial Taylor shifts.

The benchmarks are performed on polynomials with exact integer coefficients. Nevertheless, we emphasize that ANewDsc can process inputs with arbitrary approximable, possibly irrational coefficients. We observe that the performance is unaffected if, for example, the coefficients of the random instances (Tables 4 and 5) are drawn from a continuous distribution, or the coefficients of the Mignotte-like polynomials are irrational numbers of comparable magnitude.

Acknowledgments. We thank Maplesoft for their support in the development of RS, Leonardo Robol (Katholieke Universiteit Leuven) for his help and support on running the development version of MPSolve and Adam Strzebonski (Wolfram Research) for his instructions on how to control the behavior of Mathematica's `RootIntervals` function.

5. REFERENCES

[1] G. E. Collins. Continued fraction real root isolation using the Hong bound. *JSC*, 2014.

[2] G. E. Collins & A. G. Akritas. Polynomial real root isolation using Descartes' Rule of Signs. In *SYMSAC*, pp. 272–275, 1976.

[3] Z. Du, V. Sharma & C. K. Yap. Amortized Bounds for Root Isolation via Sturm Sequences. In *SNC*, pp. 113–129, 2007.

[4] A. Eigenwillig. *Real Root Isolation for Exact and Approximate Polynomials Using Descartes' Rule of Signs*. PhD thesis, Saarland University, 2008.

[5] A. Eigenwillig et al. A Descartes algorithm for polynomials with bit-stream coefficients. In *CASC*, pp. 138–149, 2005.

[6] A. Eigenwillig, V. Sharma & C. K. Yap. Almost tight complexity bounds for the Descartes method. In *ISSAC*, pp. 151–158, 2006.

[7] I. Z. Emiris, V. Y. Pan & E. P. Tsigaridas. Algebraic algorithms. In *Computing Handbook: Computer Science and Software Engineering*, ch. 10, pp. 1–30. CRC, 3rd ed., 2014.

[8] J. R. Johnson & W. Krandick. Polynomial real root isolation using approximate arithmetic. In *ISSAC*, pp. 225–232, 1997.

[9] A. Kobel & M. Sagraloff. On the complexity of computing with planar algebraic curves. *J. Compl.*, 31(2):206–236, 2014.

[10] W. Krandick. Isolierung reeller Nullstellen von Polynomen. In J. Herzberger, editor, *Wissenschaftliches Rechnen*, pp. 105–154. Akademie Verlag, Berlin, 1995.

[11] K. Mehlhorn & M. Sagraloff. A deterministic Descartes algorithm for real polynomials. *JSC*, 46(1):70–90, 2011.

[12] K. Mehlhorn, M. Sagraloff & P. Wang. From approximate factorization to root isolation with application to cylindrical algebraic decomposition. *JSC*, 66:34–69, 2015.

[13] N. Obreshkoff. *Zeros of Polynomials*. Marina Drinov, Sofia, 2003. Translation of the Bulgarian original.

[14] V. Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *JSC*, 33(5):701–733, 2002.

[15] N. Revol & F. Rouillier. Motivations for an arbitrary precision interval arithmetic and the MPFI library. *Reliable Computing*, 11(4):275–290, 2005.

[16] F. Rouillier & P. Zimmermann. Efficient isolation of [a] polynomial's real roots. *JCAM*, 162:33–50, 2004.

[17] M. Sagraloff. When Newton meets Descartes: A simple and fast algorithm to isolate the real roots of a polynomial. In *ISSAC*, pp. 297–304, 2012.

[18] M. Sagraloff. On the complexity of the Descartes method when using approximate arithmetic. *JSC*, 65(0):79–110, 2014.

[19] M. Sagraloff & K. Mehlhorn. Computing real roots of real polynomials. *JSC*, 73:46–86, 2016.

[20] E. P. Tsigaridas. Improved bounds for the CF algorithm. *TCS*, 479:120–126, 2013.

[21] C. K. Yap & M. Sagraloff. A simple but exact and efficient algorithm for complex root isolation. In *ISSAC*, pp. 353–360, 2011.