Novel Range Functions via Taylor Expansions and Recursive Lagrange Interpolation with Application to Real Root Isolation

Kai Hormann
kai.hormann@usi.ch
Università della Svizzera italiana
Lugano, Switzerland

Lucas Kania
lucas.kania@usi.ch
Università della Svizzera italiana
Lugano, Switzerland

Chee Yap
yap@cs.nyu.edu
Courant Institute, NYU
New York, USA

ABSTRACT

Range functions are an important tool for interval computations, and they can be employed for the problem of root isolation. In this paper, we first introduce two new classes of range functions for real functions. They are based on the remainder form by Cornelius and Lohner [7] and provide different improvements for the remainder part of this form. On the one hand, we use centered Taylor expansions to derive a generalization of the classical Taylor form with higher than quadratic convergence. On the other hand, we propose a recursive interpolation procedure, in particular based on quadratic Lagrange interpolation, leading to *recursive Lagrange forms* with cubic and quartic convergence. We then use these forms for isolating the real roots of square-free polynomials with the algorithm Eval, a relatively recent algorithm that has been shown to be effective and practical. Finally, we compare the performance of our new range functions against the standard Taylor form. Range functions are often compared in isolation; in contrast, our holistic comparison is based on their performance in an application. Specifically, Eval can exploit features of our recursive Lagrange forms which are not found in range functions based on Taylor expansion. Experimentally, this yields at least a twofold speedup in Eval.

CCS CONCEPTS

• **Mathematics of computing** → **Interval arithmetic**; *Computations on polynomials*; • **Computing methodologies** → *Symbolic calculus algorithms*;

KEYWORDS

range functions, root isolation, interval arithmetic

ACM Reference Format:
Kai Hormann, Lucas Kania, and Chee Yap. 2021. Novel Range Functions via Taylor Expansions and Recursive Lagrange Interpolation with Application to Real Root Isolation. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '21), July 18–22, 2021, Saint Petersburg.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ISSAC '21, July 18–22, 2021, Saint Petersburg, Russia

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This paper addresses two related computational problems: (P1) range functions and (P2) root isolation. Computing the range of functions is arguably the most basic task in interval computation [11, 19, 23]. Root isolation is also a fundamental task in the huge classical literature on root finding [17]. These two problems are connected by the fact that root isolation can be reduced to evaluating range functions. To see this, the next two subsections review the relevant literature on range functions and root isolation.

1.1 Range functions

We first consider problem (P1). Let $f\colon \mathbb{R} \to \mathbb{R}$ be a real function. For any $S \subseteq \mathbb{R}$, the *range* of f on S is the set $f(S) := \{f(x) : x \in S\}$ and we define the *magnitude* of S as $|S| := \sup\{|s| : s \in S\}$. Let $\square\mathbb{R}$ denote the set of closed bounded intervals. For any $I \in \square\mathbb{R}$ with $I = [a, b]$, the *width*, *radius*, and *midpoint* of I are given by $w(I) := b - a$, $r(I) := (b - a)/2$, and $m(I) := (a + b)/2$, respectively. Note that $|I| = \max\{|a|, |b|\}$. A *range function* (or inclusion function) for f is a function of the form

$$\square f\colon \square\mathbb{R} \to \square\mathbb{R},$$

where $f(I) \subseteq \square f(I)$ for all $I \in \square\mathbb{R}$. If $\square f(I) = f(I)$ for all I, we call it the *exact range function*. Note that '$\square f$' is a generic name for a range function of f; we use subscripts and/or superscripts to identify particular range functions: e.g., $\square_g f$, $\square_2^T f$, or $\square_3^L f$. We can compare range functions using a natural "tightness partial order" on range functions of f: we say that $\square_1 f$ is *as tight as* $\square_2 f$, denoted $\square_1 f \preceq \square_2 f$, if $\square_1 f(I) \subseteq \square_2 f(I)$ for all I. Generally, we prefer range functions that are as tight as possible, ideally the exact range function. But since tight range functions are inefficient (i.e., expensive to compute), we must choose a trade-off between tightness and efficiency. Comparative studies of range functions based on tightness or efficiency are often done in isolation, independent of any application. For example, see [8, 9, 31]. In this paper, we give a holistic or integrated comparison of range functions, namely comparisons in the context of an application (see Sec. 5).

A more robust way to evaluate range functions is to look at "asymptotic tightness". We say that $\square f$ has *order k convergence* (for $k \geq 1$) on I_0 if there exists a constant $C_0 > 0$ that depends on f and I_0 but not on I, such that

$$q(f(I), \square f(I)) \leq C_0 w(I)^k$$

for all $I \subseteq I_0$, where $q([a, b], [a', b']) := \max\{|a - a'|, |b - b'|\}$ is the Hausdorff distance on intervals. If $\square f$ has at least order 1 convergence, then we call $\square f$ *convergent*. Note that for any sequence $(I_i)_{i\geq 1}$ of intervals that converges monotonically to a point $p \in I_0$,

a convergent range function satisfies

$$f(p) = \lim_{i\to\infty} \square f(I_i).$$

Such a convergent range function is also called a *box form* of f [31]. When $k = 2$, we say $\square f$ has quadratic convergence.

Cornelius and Lohner [7] were the first to introduce techniques for higher than quadratic convergence. For any function $g\colon \mathbb{R} \to \mathbb{R}$, they consider range functions of f of the form

$$\square_g f(I) := g(I) + \square R_g(I), \tag{1}$$

where $R_g := f - g$ is the remainder function. They call g the *exact part* of this range function because its range must be computed exactly. This limits g to polynomials of small degree d (Cornelius and Lohner suggest $d \le 5$). The *remainder part* $\square R_g(I)$ need not be exact, but its width controls the overall Hausdorff distance, since [7, Theorem 4]

$$q(f(I), \square_g f(I)) \le w(\square R_g(I)).$$

It follows that the *remainder form* $\square_g f(I)$ has order k convergence, if $w(\square R_g(I)) \le C_0 w(I)^k$.

Cornelius and Lohner show that this can be achieved by letting the exact part g be a Hermite interpolant of f. In fact, if f is k times continuously differentiable, $x_0, \dots, x_\ell \in I$ are distinct interpolation nodes, $p_0, \dots, p_\ell$ are positive integers with $\sum_{i=0}^{\ell} p_i = k$, and g is the unique polynomial of degree at most $k-1$, such that

$$g^{(j)}(x_i) = f^{(j)}(x_i), \qquad j = 0, \dots, p_i - 1, \quad i = 0, \dots, \ell, \tag{2}$$

then the remainder function can be expressed for any $x \in I$ as

$$R_g(x) = \frac{1}{k!} f^{(k)}(\xi_x) \prod_{i=0}^{\ell} (x - x_i)^{p_i}, \tag{3}$$

for some $\xi_x \in I$. We now define the remainder part as

$$\square R_g(I) := \frac{1}{k!} \square f^{(k)}(I) \prod_{i=0}^{\ell} (I - x_i)^{p_i}, \tag{4}$$

where $\square f^{(k)}(I)$ is what Ratschek and Rokne [23, p. 23] call the *natural interval extension* of $f^{(k)}(x)$. For example, if $f^{(k)}(x)$ is a polynomial, we write it as an expression $E(x)$ in the nested Horner form and define $\square f^{(k)}(I) := E(I)$. The remainder form $\square_g f(I)$ in (1) then has order k convergence, because $|I - x_i| \le w(I)$ and Lemma 1.6 in [23, p. 24] imply

$$w(\square R_g(I)) \le 2|\square R_g(I)| \le 2\frac{|\square f^{(k)}(I)|}{k!} w(I)^k \le 2\frac{|\square f^{(k)}(I_0)|}{k!} w(I)^k.$$

The simplest example of this approach is the convergent *mean value form* around x_0,

$$\square^{MV}_{x_0} f(I) := f(x_0) + \square f'(I)(I - x_0),$$

which is obtained by letting $\ell = 0$ and $p_0 = k = 1$, so that g is the constant interpolant of f at x_0. This form has even quadratic convergence, if the range $f'(I)$ is approximated with a Lipschitz range function [23].

Cornelius and Lohner further point out that it is also possible to define the exact part as

$$\hat{g}(x) := g(x) + \frac{y}{k!} \prod_{i=0}^{\ell} (x - x_i)^{p_i} \tag{5}$$

for some $y \in f^{(k)}(I) \subset \square f^{(k)}(I)$ and the remainder part (cf. (4)) as

$$\square R_{\hat{g}}(I) := \frac{1}{k!} (\square f^{(k)}(I) - y) \prod_{i=0}^{\ell} (I - x_i)^{p_i}. \tag{6}$$

If $f^{(k)}$ is Lipschitz continuous, then this gives one extra order of convergence, because $|\square f^{(k)}(I) - y| \le w(\square f^{(k)}(I)) \le C_0' w(I)$ for all $I \subseteq I_0$ and some constant $C_0' > 0$ that depends on f and I_0 but not on I [7, Theorem 2]. In this variant, $\hat{g}$ is of degree k and the condition that distinguishes $\hat{g}$ from g is that $\hat{g}^{(k)} = y$, while $g^{(k)} = 0$. Evaluating $\hat{g}(I)$ exactly is of course more costly than evaluating $g(I)$, because $\hat{g}$ has a higher degree than g. Note that we can also get this extra order of convergence by adding one Hermite interpolation condition to the definition of g. The evaluation of the exact part would then be as costly as the evaluation of $\hat{g}(I)$, but the remainder part would depend on $f^{(k+1)}$, while the remainder part in (6) depends on $f^{(k)}$, a fact that we shall exploit in Sec. 3.2.

The Cornelius–Lohner framework appears to suggest that convergence is limited by the exact part alone, without attaching much interest to the remainder part. In this paper, we suggest the contrary: for any function f with exact part g, the remainder part $\square R_g(I)$ in (1) can vary. Despite having the same order of convergence, their actual performance in an application like root isolation can diverge significantly.

In this paper, we propose two new ideas for defining such improved remainder parts. The first relies on expressing the remainder function (3) in centered form (Sec. 2.1), the second approximates $f^{(k)}(I)$ in (4) using again the remainder form in (1), thus applying the idea of Cornelius and Lohner recursively (Sec. 3).

1.2 Real root isolation and Eval

We next turn to (P2). Consider again a real function $f\colon \mathbb{R} \to \mathbb{R}$. The *zero set* of f on $S \subseteq \mathbb{R}$ is $\mathtt{Zero}_f(S) := \{x \in S : f(x) = 0\}$, and $\#_f(S)$ denotes[1] the cardinality of $\mathtt{Zero}_f(S)$. An *isolator* for f is an interval I such that $\#_f(I) = 1$, and we say that I *isolates* the unique zero of f in I. The root isolation problem can then be formalized as follows: Given f and an interval $I_0 \in \square\mathbb{R}$, compute a set Z of isolators for f, such that each $\zeta \in \mathtt{Zero}_f(I_0)$ is isolated by some $I \in Z$. Assuming f to be *nice*, in the sense that f is continuously differentiable and the zeros of f in I_0 are *simple* (i.e., $f(\zeta) = 0$ implies $f'(\zeta) \ne 0$), we can reduce problem (P2) to (P1) using a procedure that we call Eval (see Algo. 1).

Note that the numerical computation of Eval is reduced to evaluating two range functions, one for f (line 5) and one for its derivative f' (line 6). Moreover, Eval uses two queues to hold intervals, an *active queue* Q and an *output queue* Z. The intervals I are bisected until either $0 \notin \square f(I)$ or $0 \notin \square f'(I)$ holds. We may call these two conditions the exclusion and inclusion predicates.

Eval terminates and solves problem (P2), if we assume the two range functions $\square f$ and $\square f'$ to be convergent on I_0. It is then clear that each $I \in Z$ represents a unique root $\zeta \in \mathtt{Zero}_f(I_0)$, because I is added to Z if and only if $f(a)f(b) \le 0$ (line 9), which guarantees the existence of a root by the intermediate value theorem, and if f is

[1] Note that root multiplicity is not used in the definitions of $\mathtt{Zero}_f(S)$ and $\#_f(S)$. In particular, $\mathtt{Zero}_f(S)$ is a set, not a multiset.

Algorithm 1 Real root isolation with range functions

Input: $f : \mathbb{R} \to \mathbb{R}$ and $I_0 \in \square\mathbb{R}$
Output: Z containing isolators for each $\zeta \in \mathtt{Zero}_f(I_0)$

```
1:  procedure Eval(f, I_0)
2:      initialize Q := {I_0} and Z := ∅
3:      while Q is non-empty do
4:          I := Q.pop(), where I = [a, b]
5:          if 0 ∈ □f(I) then          ▷ I is implicitly discarded if 0 ∉ □f(I)
6:              if 0 ∈ □f'(I) then
7:                  Q.push([a, m], [m, b]), where m = m(I)
8:              else                                  ▷ f is strictly monotonic
9:                  if f(a)f(b) ≤ 0 then                          ▷ 0 ∈ f(I)
10:                     Z.push(I)
11:     return Z
```

strictly monotonic on $I = [a, b]$ (line 8), which assures the uniqueness of that root. Moreover, each $\zeta \in \mathtt{Zero}_f(I_0)$ is represented by at most two isolators. In case two isolators $I, J \in Z$ represent ζ, then $\zeta \in I \cap J$ is a common endpoint of I and J. Such duplication is easily detected and removed, or avoided upfront. For example, if f is a polynomial with rational coefficients and rational arithmetic is used in Eval, then we can replace the weak inequality in line 9 by the strict inequality $f(a)f(b) < 0$ and instead test $f(m) = 0$ after line 7, adding the point interval $[m, m]$ to Z if the test holds.

Despite its simplicity, the subdivision tree of Eval is "near-optimal" when f is an integer polynomial [3, 4, 28] and the box forms $\square f$ and $\square f'$ are the "maximal" centered Taylor forms $\square_2^T$ (see Sec. 2). In other words, it asymptotically matches the tree size achieved by powerful tools like Sturm sequences or Descartes' rule of signs! However, Eval does not require f to be a polynomial [32].

1.3 Some broader literature

Besides the book of Ratschek and Rokne [23] on range functions, we refer to Neumaier [20, Chapter 2.4] and Stahl's thesis [29] for further investigations of the remainder forms of Cornelius and Lohner [7], which are also referred to as *interpolations forms*.

To our knowledge, the first version of Eval is from Mitchell [18] in the context of ray tracing in computer graphics. Its current formulation as a root isolation algorithm, together with complexity analysis, began with [5]. Yap et al. introduced Eval as a 1-dimensional analogue of the 2-dimensional algorithm of Plantinga and Vegter for isotopic approximation of non-singular curves [14, 15, 22]. Besides Eval, Yap et al. also introduced CEval [25] for complex roots, and AEval [32] for analytic roots. The complexity analysis of these algorithms can be captured under the elegant framework of "continuous amortization" [3, 4, 28].

Root finding for polynomials is a highly classical problem [16, 17] that has remained active to the present. The modern complexity-theoretic approach to root finding was initiated by Schönhage in 1982 [26]. A basic quest is to construct "near-optimal" algorithms, and in the last decade, significant progress has been made in this direction; see Sagraloff and Mehlhorn [24] (for real roots) and Becker et al. [1, 2] (for complex roots). The new near-optimal algorithms (like Eval) are based on the subdivision paradigm; moreover, they were implemented soon after their appearance [12, 13]. In contrast, the original near-optimal algorithm [21] has never been implemented (see [21, p. 703] for some challenges).

1.4 Overview of the paper

In Section 2, we introduce a family of range functions based on Taylor expansions. Technically, these functions are not new, but within the Cornelius–Lohner framework, we highlight their true role as improvements on the remainder parts. In Section 3, we introduce range functions based on recursive Lagrange interpolation. These are new but again, we can view them as improvements of the remainder parts. In Secs. 4 and 5, we evaluate the deployment of eight of these range functions in the Eval algorithm; here, the Lagrange form begins to shine because of its "distributed evaluation" scheme (see Sec. 4.1). We conclude in Sec. 6. **NOTE:** This version refers to appendices (A.1–A.3) for the benefit of the reviewers. These three appendices are easily removed in the final version, with trivial modifications and bringing the page count down to 8.

2 NEW RANGE FUNCTIONS BASED ON CENTERED TAYLOR EXPANSIONS

A classic approach for designing a remainder form (1) with quadratic convergence is to choose $\ell = 0$ and $p_0 = k = 2$ in (2) and letting $x_0 = m := m(I)$, so that the exact part is the *linear* Taylor polynomial of f about the midpoint of I, that is, $g_1(x) := f(m)+(x-m)f'(m)$. This gives the *centered form* $\square_{g_1} f(I) := g_1(I) + \square R_{g_1}(I)$. One option now is to follow Cornelius and Lohner and express the remainder part as in (4),

$$\square R_{g_1}(I) = \frac{1}{2}\square f''(I)(I-m)^2, \tag{7}$$

where $\square f''(I)$ is the natural interval extension of $f''(x)$. We call the resulting version of $\square_{g_1} f(I)$ the *minimal* (centered) Taylor form.

This can be improved considerably, if f is n times continuously differentiable for $n > 2$, by using the $(n-1)$-th order Taylor expansion of f about m to write the remainder function as

$$R_{g_1}(x) = \sum_{i=2}^{n-1} \frac{f^{(i)}(m)}{i!}(x-m)^i + \frac{f^{(n)}(\xi_x)}{n!}(x-m)^n, \tag{8}$$

for some $\xi_x \in I$. We now define

$$c_i := \frac{f^{(i)}(m)}{i!}, \quad i = 0, \ldots, n-1, \qquad c_n := \frac{|\square f^{(n)}(I)|}{n!}, \tag{9}$$

where the magnitude of the natural interval extension, $\square f^{(n)}(I)$, can be replaced by $f^{(n)}(m)$ in the definition of c_n, if $f^{(n)}$ is a constant, for example, in the case of f being a polynomial of degree $d \le n$. We then get the following improvement of (7):

$$\square R_{g_1}(I) := \sum_{i=2}^{n} c_i (I-m)^i = r^2[-1,1]S_{2,n}, \qquad S_{2,n} := \sum_{i=2}^{n} |c_i| r^{i-2}, \tag{10}$$

where $r := r(I)$. Computing the c_i's takes $O(n \log n)$ arithmetic steps (or $O(n^2)$ in simple implementations, as in Sec. 5); for bit-complexity, see [30]. In contrast, the natural interval extension (7) requires $O(n)$ steps. What do we get in return? Although this does not change the quadratic convergence of the centered form $\square_{g_1} f(I)$, it may be much better than the remainder part in (7) of Cornelius and Lohner, because successive terms of $S_{2,n}$ converge with higher and higher order. This is dramatically illustrated below in Tables 2–4 (columns $\widetilde{\mathrm{E}}_2^T$ and E_2^T). Recalling that the *exact* range of g_1 is

$g_1(I) = c_0 + r[-1,1]c_1$ (see App. A.1), we realize that the resulting centered form

$$\square^T_{2,n} f(I) := c_0 + r[-1,1]c_1 + r^2[-1,1]S_{2,n} \tag{11}$$

is actually just the classical *Taylor form* of order n (or "level n" using our terminology below) [23, p. 77], with the range $f^{(n)}(I)$ approximated by $|\square f^{(n)}(I)| \cdot [-1,1]$.

2.1 Taylor forms with order k convergence

Following Cornelius and Lohner, we can raise the convergence order from quadratic to basically any order $k > 2$, simply by replacing g_1 with the $(k-1)$-th order Taylor polynomial of f about m,

$$g_{k-1}(x) := \sum_{i=0}^{k-1} \frac{f^{(i)}(m)}{i!}(x-m)^i = \sum_{i=0}^{k-1} c_i (x-m)^i. \tag{12}$$

But instead of expressing the remainder function $R_{g_k} = f - g_k$ in terms of the k-th derivative of f as (cf. (3))

$$R_{g_{k-1}}(x) = \frac{1}{k!} f^{(k)}(\xi_x)(x-m)^k,$$

we continue the Taylor expansion of $f^{(k)}(x)$ all the way to $n-1$ for some $n \geq k$ (assuming that the derivatives exist), to obtain (cf. (8))

$$R_{g_{k-1}}(x) = \sum_{i=k}^{n-1} \frac{f^{(i)}(m)}{i!}(x-m)^i + \frac{f^{(n)}(\xi_x)}{n!}(x-m)^n, \tag{13}$$

for some $\xi_x \in I$. As above (cf. (10) and (11)), we then get the *generalized Taylor form* of (convergence) *order* k and *level* n:

$$\square^T_{k,n} f(I) := g_{k-1}(I) + r^k[-1,1]S_{k,n}, \qquad S_{k,n} := \sum_{i=k}^{n} |c_i| r^{i-k}, \tag{14}$$

where the c_i are defined as in (9). The level n is *minimal* if $n = k$, and *maximal* if $n = \infty$. The maximal level is only possible when f is analytic and r sufficiently small, so that $S_{k,\infty}$ is convergent. Clearly, if f is a polynomial of degree d, then $S_{k,\infty}$ is a finite sum and convergent for any r. We call the corresponding range functions *minimal* and *maximal* Taylor forms of order k, denoted by $\widetilde{\square}^T_k f(I)$ and $\square^T_k f(I)$, respectively. This definition includes the minimal Taylor form based on g_1 (cf. (7)) as a special case for $k = n = 2$.

For $k = 3$, computing the exact range of the quadratic Taylor polynomial g_2 is only marginally more costly (see App. A.2) than computing $g_1(I)$ and the cubic convergence gives a noticeable performance gain when used in Eval (see Sec. 5). But already for $k = 4$ the computational overhead of determining the range $g_3(I)$ exactly (see App. A.3) appears to outweigh the benefit of the better convergence order, at least in the context of Eval, leaving only a slight advantage in terms of running time. Note that there is a similar phenomenon in Newton's method where quadratic convergence is the sweet spot despite the possibility of achieving cubic (Halley's method) or higher convergence.

3 NEW RANGE FUNCTIONS BASED ON RECURSIVE INTERPOLATION

Another approach to improving the remainder part is by recursively applying the idea of Cornelius and Lohner. To this end, let h_0 be the Hermite interpolant of f for a certain choice of interpolation nodes x_i and multiplicities p_i and with degree at most $k-1$. According to (3), the remainder part $R_{h_0} = f - h_0$ can be written as

$$R_{h_0}(x) = \frac{\omega(x)}{k!} f^{(k)}(\xi_x), \qquad \omega(x) := \prod_{i=0}^{\ell} (x - x_i)^{p_i}, \tag{15}$$

for some $\xi_x \in I$, and the magnitude of its (exact) range satisfies

$$|R_{h_0}(I)| \leq \Omega |f^{(k)}(I)|, \qquad \Omega := \frac{|\omega(I)|}{k!}. \tag{16}$$

Here we assume that the range $\omega(I)$ can be computed exactly, which is certainly true for small k (as in Sec. 3.1 below), but it is also possible to replace $\omega(I)$ with some range estimate $\square\omega(I)$. We now split $f^{(k)}$ in (16) into the Hermite interpolant h_1 of $f^{(k)}$ (for the same interpolation nodes and multiplicities) and a remainder part R_{h_1}. Since $|R_{h_1}(I)| \leq \Omega |f^{(2k)}(I)|$, we obtain

$$|f^{(k)}(I)| \leq |h_1(I)| + \Omega |f^{(2k)}(I)|. \tag{17}$$

If f is nk times continuously differentiable for some $n \geq 1$, we may repeat this procedure (always with the same interpolation nodes x_i and multiplicities p_i) to obtain Hermite interpolants h_j of $f^{(jk)}$ for $j = 1, \ldots, n$. This gives a recursive remainder bound

$$|R_{h_0}(I)| \leq \sum_{j=1}^{n-1} |h_j(I)| \Omega^j + \Omega^n |\square f^{(nk)}(I)| =: T_{k,n}. \tag{18}$$

Since $\omega(x)$ scales with r^k as I varies, we have $\Omega \in O(r^k)$ and also $T_{k,n} \in O(r^k)$. It follows that the *recursive remainder form* of *order* k and *level* n,

$$\square^R_{k,n} f(I) := h_0(I) + [-1,1]T_{k,n}, \tag{19}$$

has indeed order k convergence. The *minimal* form $\square^R_{k,1} f(I)$ for the smallest level $n = 1$ is essentially the remainder form of Cornelius and Lohner (cf. (4)), if we replace $\omega(I)$ in (16) by $\square\omega(I)$. As in Sec. 2, the advantage of higher levels of n is due to the fact that the terms of $T_{k,n}$ converge with successively higher order. Again, the maximal level $n = \infty$ that induces the *maximal* recursive remainder form $\square^R_{k,\infty} f(I)$, is only possible if T_∞ is convergent, which is the case if f is analytic and r sufficiently small, or if f is a polynomial. Note that in the latter case, evaluating this form requires just a finite number of point evaluations of f and its derivatives, akin to the evaluation of the maximal Taylor forms.

3.1 Recursive Lagrange form with cubic convergence

One particular instance of the recursive remainder form (19) that will prove beneficial for Eval is based on the endpoints and the midpoint of $I = [a,b]$ as simple interpolation nodes, that is, to use $\ell = 2$, $x_0 = a$, $x_1 = m$, $x_2 = b$ in (2) and $p_0 = p_1 = p_2 = 1$, so that $k = 3$. In this setting, h_j is the quadratic *Lagrange interpolant* of $f^{(3j)}$ at a, m, and b, which can be expressed in centered form as

$$h^L_j(x) := d_{j,0} + d_{j,1}(x-m) + d_{j,2}(x-m)^2 \tag{20}$$

with coefficients

$$d_{j,0} := f^{(3j)}(m), \qquad d_{j,1} := \frac{f^{(3j)}(b) - f^{(3j)}(a)}{2r},$$

$$d_{j,2} = \frac{f^{(3j)}(b) - 2f^{(3j)}(m) + f^{(3j)}(a)}{2r^2},$$

where $r := r(I)$. A simple calculation shows that the exact range of

$$\omega_3(x) := (x-a)(x-m)(x-b)$$

is $\omega_3(I) = \frac{2\sqrt{3}}{9} r^3 [-1,1]$, so that $\Omega_3 := \frac{1}{6}|\omega_3(I)| = \frac{\sqrt{3}}{27} r^3$. We denote the resulting *recursive Lagrange form* of *level n* by

$$\square^L_{3,n} f(I) := h^L_0(I) + [-1,1] T_{3,n}, \tag{21}$$

where (cf. (18))

$$T_{3,n} := \sum_{j=1}^{n-1} |h^L_j(I)| \Omega_3^j + \Omega_3^n |\square f^{(3n)}(I)| \in O(r^3). \tag{22}$$

If f is a polynomial of degree d, then the *maximal* recursive Lagrange form $\square^L_3 f(I) := \square^L_{3,\infty} f(I)$ depends on the $3(\lfloor d/3 \rfloor + 1)$ values $f^{(3j)}(a)$, $f^{(3j)}(m)$, $f^{(3j)}(b)$, $j = 0, \ldots, \lfloor d/3 \rfloor$, which is comparable to the $d+1$ values needed for the maximal Taylor forms $\square^T_k f(I)$.

As the cubic convergence of $\square^L_{3,n} f(I)$ is *independent* of how the range of h^L_j is estimated for $j \geq 1$ in (22), we can replace the exact evaluation of $h^L_j(I)$ by the cheaper centered form evaluation

$$\square^T_2 h^L_j(I) = d_{j,0} + r[-1,1]|d_{j,1}| + r^2[-1,1]|d_{j,2}|.$$

This yields a less tight range function (cf. (21))

$$\square^{L'}_{3,n} f(I) := h^L_0(I) + [-1,1] T'_{3,n}, \tag{23}$$

where

$$T'_{3,n} := \sum_{j=1}^{n-1} \big(|d_{j,0}| + r|d_{j,1}| + r^2|d_{j,2}|\big)\Omega_3^j + \Omega_3^n |\square f^{(3n)}(I)| \in O(r^3),$$

which depends on the same data values as $\square^L_{3,n} f(I)$. In the context of Eval, this increases the size of the subdivision tree slightly, but seems to be more efficient in terms of running time (see Sec. 5).

3.2 Recursive Lagrange form with quartic convergence

Another variant of the recursive Lagrange form can be obtained by applying Cornelius and Lohner's general trick to get one extra order of convergence. To this end (cf. (20)), let

$$\begin{aligned} \hat{h}^L_0(x) &:= h^L_0(x) + \frac{f'''(m)}{6}\omega_3(x) \\ &= d_{0,0} + \hat{d}_{0,1}(x-m) + d_{0,2}(x-m)^2 + \hat{d}_{0,3}(x-m)^3, \end{aligned} \tag{24}$$

where

$$\hat{d}_{0,1} := d_{0,1} - r^2 \frac{f'''(m)}{6}, \qquad \hat{d}_{0,3} := \frac{f'''(m)}{6},$$

be the (unique) cubic polynomial that interpolates f at a, m, and b, like h^L_0, and also matches the third derivative of f at m, in the sense that $(\hat{h}^L_0)'''(m) = f'''(m)$. Similarly as above, we then have

$$|R_{\hat{h}^L_0}(I)| \leq \Omega_3 |f'''(I) - f'''(m)| = \Omega_3 |\hat{f}_3(I)|,$$

where $\hat{f}_3(x) := f'''(x) - f'''(m)$. We now split $\hat{f}_3$ into the Lagrange interpolant

$$\hat{h}^L_1(x) := h^L_1(x) - f'''(m) = (d_{1,1} + d_{1,2}(x-m))(x-m) \tag{25}$$

of $\hat{f}_3$ at a, m, and b and the remainder $R_{\hat{h}^L_1}$, which satisfies

$$|R_{\hat{h}^L_1}(I)| \leq \Omega_3 |\hat{f}'''_3(I)| = \Omega_3 |f^{(6)}(I)|,$$

hence $|\hat{f}_3(I)| \leq |\hat{h}^L_1(I)| + \Omega_3 |f^{(6)}(I)|$. From here on, we repeat the splitting procedure as in the construction of $\square^L_{3,n} f(I)$ and finally arrive at the recursive Lagrange form of level n

$$\square^L_{4,n} f(I) := \hat{h}^L_0(I) + [-1,1] T_{4,n}, \tag{26}$$

where (cf. (22))

$$T_{4,n} := |\hat{h}^L_1(I)|\Omega_3 + \sum_{j=2}^{n-1} |h^L_j(I)|\Omega_3^j + \Omega_3^n |\square f^{(3n)}(I)|.$$

The advantage of $\square^L_{4,n} f(I)$ in (26) over $\square^L_{3,n} f(I)$ in (21) is that $|\hat{h}^L_1(I)| \in O(r)$, which follows from (25), so that $T_{4,n} \in O(r^4)$. This implies that $\square^L_{4,n} f(I)$ has *quartic* convergence, at the cost of requiring the evaluation of the exact range of the cubic polynomial $\hat{h}^L_0$ in (24).

Note that $\square^L_{4,n} f(I)$ depends on the same data as $\square^L_{3,n} f(I)$, and analogous to (23), we can replace the exact evaluation of $\hat{h}^L_1(I)$ and $h^L_j(I)$ for $j \geq 2$ by centered form evaluations to get the cheaper, but less tight range function

$$\square^{L'}_{4,n} f(I) := \hat{h}^L_0(I) + [-1,1] T'_{4,n}, \tag{27}$$

where $T'_{4,n} := T'_{3,n} - |d_{1,0}|\Omega_3$, without compromising the quartic convergence order, because also $T'_{4,n} \in O(r^4)$.

A valid question at this point is: why did we not consider applying Cornelius and Lohner's trick for increasing the convergence order to the generalized Taylor forms in Sec. 2.1? The answer is surprisingly simple: because it does not give anything new! In fact, if we modify the exact part $g_{k-1}(x)$ of the Taylor form $\square^T_{k,n} f(I)$ accordingly and consider the alternative exact part $\hat{g}_{k-1}(x) := g_{k-1}(x) + \frac{f^{(k)}(m)}{k!}(x-m)^k$, then we eventually get the Taylor form $\square^T_{k+1,n} f(I)$, because $\hat{g}_{k-1} = g_k$.

4 REAL ROOT ISOLATION WITH EVAL AND THE NEW RANGE FUNCTIONS

4.1 Advantage of the Lagrange form in Eval

What is to recommend the generalized Taylor form or the recursive Lagrange form? We give the intuition for the advantages of the Lagrange form *in the context of root isolation with* Eval for polynomials. Recall that computing the maximal Taylor form $\square^T_k f(I)$ for a polynomial of degree d requires us to evaluate $f^{(i)}$ at $m = m(I)$ for $i = 0, \ldots, d$. To compute the maximal recursive Lagrange form $\square^L_3 f(I)$, we must evaluate $f^{(3j)}$ at a, m, b, where $I = [a,b]$ for $j = 0, \ldots, \lfloor d/3 \rfloor$. Considered in isolation, the two forms are comparable in computational complexity, since they each need about d function or derivative evaluations. But in the context of the Eval algorithm, the Lagrange form begins to shine: after estimating the range of f over $[a,b]$, we would typically need to further estimate the ranges over $[a,m]$ and $[m,b]$. For the Lagrange form, estimating the range over $[a,m]$ needs only $\lfloor d/3 \rfloor + 1$ additional evaluations of $f^{(3j)}$ at $(a+m)/2$, since we already computed $f^{(3j)}(a)$ and $f^{(3j)}(m)$. In contrast, the Taylor form must still make $d+1$ evaluations of f and its derivatives at $(a+m)/2$. A similar remark holds for $[m,b]$. Therefore, we may expect a roughly 3-fold speed up of Eval when using the Lagrange instead of the Taylor form, although

Table 1: Combinations of range functions for f and f' used by Eval in our experiments.

	Taylor forms				recursive Lagrange forms			
	$\widetilde{\mathrm{E}}_2^T$	E_2^T	E_3^T	E_4^T	E_3^L	$\mathrm{E}_3^{L'}$	E_4^L	$\mathrm{E}_4^{L'}$
range of f	$\widetilde{\square}_2^T$	$\square_2^T$	$\square_3^T$	$\square_4^T$	$\square_3^L$	$\square_3^{L'}$	$\square_4^L$	$\square_4^{L'}$
range of f'	$\widetilde{\square}_2^T$	$\square_2^T$	$\square_3^T$	$\square_4^T$	$\square_2^L$	$\square_2^{L'}$	$\square_2^L$	$\square_2^{L'}$

we should keep in mind that the performance is also influenced by other factors. For example, the tightness of the two forms is not identical and the Lagrange form requires a more elaborate memory management so that some of the data needed for processing $[a, m]$ and $[m, b]$ can be inherited from the data computed for $[a, b]$.

4.2 Range functions for derivatives

Before presenting the results of our numerical experiments, there is one more issue that needs to be dealt with: Eval not only needs to estimate the range of f over I, but also the range of f'.

For the generalized Taylor form, a simple calculation shows that the generalized Taylor form (of level $n-1$) applied to f' is

$$\square_{k,n-1}^T f'(I) = g_k'(I) + r^k[-1,1]S'_{k,n-1}, \quad S'_{k,n-1} := \sum_{i=k+1}^{n} i|c_i|r^{i-k-1},$$

where g_k is the k-th order Taylor polynomial of f about m, that is, $g_k'(x) = \sum_{i=1}^{k} i\,c_i(x-m)^{i-1}$, and the c_i are defined as in (9). Therefore, $\square_{k,n}^T f(I)$ and $\square_{k,n-1}^T f'(I)$ both have order k convergence and depend on the same data.

For the Lagrange form, it is more complicated, since $\square_{3,n}^L f'(I)$ depends on the evaluation of $f^{(3j+1)}$ at a, m, and b and would thus double the computational cost. To re-use the data needed for computing $\square_{3,n}^L f(I)$, we recall a result by Shadrin [27], which asserts that the error between the k-th derivative of f and the k-th derivative of the Lagrange polynomial $h(x)$ that interpolates f at the $\ell+1$ nodes $x_0, \dots, x_\ell \in I$ satisfies

$$|f^{(k)}(x) - h^{(k)}(x)| \le |\omega^{(k)}(I)| \frac{|f^{(\ell+1)}(I)|}{(\ell+1)!}, \qquad x \in I,$$

for $k = 0, \dots, \ell$ and $\omega(x) = \prod_{i=0}^{\ell}(x - x_i)$. In the context of $\square_{3,n}^L$, this implies

$$|f'(x) - (h_0^L)'(x)| \le |\omega_3'(I)| \frac{|f'''(I)|}{6}, \qquad x \in I.$$

Since $\omega_3'(I) = r^2[-1,2]$ and $\Omega_3|f'''(I)| \le T_{3,n}$, we conclude that $f'(I)$ can be estimated by the recursive Lagrange forms

$$\square_{2,n}^L f'(I) := (h_0^L)'(I) + \frac{3\sqrt{3}}{r}[-1,1]T_{3,n}, \tag{28}$$

and

$$\square_{2,n}^{L'} f'(I) := (h_0^L)'(I) + \frac{3\sqrt{3}}{r}[-1,1]T'_{3,n}, \tag{29}$$

which have only quadratic convergence, but depend on the same data as $\square_{3,n}^L f(I)$ and $\square_{3,n}^{L'} f(I)$. Note that we cannot derive a similar range function for f' with cubic convergence from $\square_{4,n}^L$, because $\hat{h}_0^L$ is *not* a Lagrange interpolant and Shadrin's result does not apply.

Table 2: Size of the Eval subdivision tree.

f	I_0	$\widetilde{\mathrm{E}}_2^T$	E_2^T	E_3^T	E_3^L	$\mathrm{E}_3^{L'}$	E_4^T	E_4^L	$\mathrm{E}_4^{L'}$
T_{20}		931	319	211	239	243	195	227	231
T_{40}		183115	663	439	471	479	423	455	463
T_{80}	$[-10,10]$	—	1379	931	983	1007	863	931	955
T_{160}		—	2751	1859	1943	1979	1723	1875	1899
T_{320}		—	5611	3795	3875	4003	3467	3735	3851
H_{20}		491	259	179	195	195	151	191	191
H_{40}		18039	443	319	359	363	303	347	351
H_{80}	$[-25,25]$	—	851	639	683	695	547	671	683
H_{160}		—	1319	1063	1123	1131	1011	1111	1119
H_{320}		—	2251	1967	1975	2063	1527	1939	1987
M_{21}		3873	169	97	113	113	91	109	109
M_{41}		—	339	181	215	215	181	213	213
M_{81}	$[-1,1]$	—	683	367	445	445	359	423	423
M_{161}		—	1379	757	905	905	721	857	857
M_{321}		—	2771	1513	1801	1801	1459	1711	1711
S_{100}		629	973	521	633	633	509	609	609
S_{200}		1251	1941	1045	1281	1281	1019	1221	1221
S_{400}	$[-10,10]$	2503	3887	2083	2555	2555	2035	2435	2435
S_{800}		5005	7753	4161	5103	5103	4053	4875	4875

5 NUMERICAL EXPERIMENTS

We implemented a general version of the Eval procedure (see Algo. 1) in *C++* and derived from it eight versions (see Table 1) that differ by the concrete range functions used for estimating the ranges of f and f' in lines 5 and 6. The first version $\widetilde{\mathrm{E}}_2^T$ estimates both ranges with the minimal Taylor form (cf. (7) in Sec. 2). The next three versions E_k^T for $k = 2, 3, 4$ employ the order-k convergent Taylor form for both ranges (see Sec. 2.1). The remaining four versions use recursive Lagrange forms with cubic or quartic convergence (see Secs. 3.1 and 3.2) to estimate the range of f and the recursive Lagrange form with quadratic convergence (see Sec. 4.2) for f'. Note that the version E_2^T represents the state-of-the-art of Eval [5] and serves as the "baseline" for performance. Except for $\widetilde{\square}_2^T$, all these Taylor and Lagrange forms are the maximal versions.

The input data for our experiments come from four *representative* families of integer polynomials: dense with all roots real (Chebyshev, T_n and Hermite, H_n), dense with only 2 real roots in I_0 (Mignotte cluster, $M_{2k+1} = x^{2k+1} - 2(4x^2-1)^k$, from [12]) and sparse without real roots ($S_n(x) = 1 + x + \sum_{i=0}^{\log_2 \frac{n}{100}} x^{2^i 100}$). Note that these polynomials do not have multiple roots, a prerequisite for Eval's halting. Our implementation, including these data and experiments, may be downloaded from the *Core Library* webpage [6, 33].

We summarize the results of our experiments in three tables, with columns grouped by convergence order. Table 2 reports the size of the Eval subdivision tree for the various polynomials. It is a good measure of the tightness of the various range functions, since the size of the recursion tree is inversely proportional to the tightness of the range functions used. In each row, we underscore the smallest tree size, which is always achieved by E_4^T. In general, we observe that the tree size decreases as the convergence order of the range functions increases and that the "cheaper" variants of the recursive Lagrange forms lead to (slightly) larger subdivision trees. The difference between the tree sizes for the Taylor and Lagrange

Table 3: Average running time of the Eval algorithm with 1024-bit floating point arithmetic in seconds.

f	I_0	$\widetilde{\mathrm{E}}_2^T$	E_2^T	E_3^T	E_3^L	$\mathrm{E}_3^{L'}$	E_4^T	E_4^L	$\mathrm{E}_4^{L'}$	σ
T_{20}		0.1242	0.02161	0.01526	0.01457	0.01200	0.01459	0.01496	0.01208	1.80
T_{40}		69.96	0.1470	0.0996	0.0677	0.0549	0.0987	0.0689	0.0555	2.68
T_{80}	$[-10, 10]$	—	1.173	0.775	0.379	0.328	0.725	0.365	0.315	3.58
T_{160}		—	9.43	6.39	2.48	2.29	5.80	2.42	2.22	4.12
T_{320}		—	77.2	52.5	17.7	17.3	48.4	17.0	16.7	4.46
H_{20}		0.06296	0.01762	0.01283	0.01214	0.01022	0.01167	0.01271	0.01014	1.72
H_{40}		6.263	0.0945	0.0685	0.0499	0.0403	0.0679	0.0505	0.0412	2.34
H_{80}	$[-25, 25]$	—	0.706	0.528	0.258	0.223	0.450	0.259	0.222	3.17
H_{160}		—	4.40	3.54	1.46	1.31	3.40	1.41	1.28	3.36
H_{320}		—	31.5	27.0	8.9	8.8	21.1	8.8	8.5	3.58
M_{21}		0.5314	0.01389	0.007585	0.007525	0.005753	0.006891	0.007448	0.005920	2.41
M_{41}		—	0.07723	0.04097	0.03071	0.02430	0.04075	0.03071	0.02376	3.18
M_{81}	$[-1, 1]$	—	0.5599	0.3020	0.1681	0.1409	0.2940	0.1624	0.1376	3.97
M_{161}		—	4.620	2.507	1.152	1.049	2.403	1.094	0.9977	4.41
M_{321}		—	38.52	21.08	8.247	7.842	20.47	7.883	7.449	4.91
S_{100}		0.8973	1.080	0.582	0.346	0.301	0.572	0.336	0.292	3.59
S_{200}	$[-10, 10]$	6.124	8.54	4.62	2.27	2.09	4.50	2.19	2.00	4.09
S_{400}		47.22	66.9	36.3	16.2	15.4	35.2	15.4	14.7	4.34
S_{800}		368.3	527	281	120	117	273	113	112	4.50

Table 4: Average running time of the Eval algorithm with multi-precision rational arithmetic in seconds.

f	I_0	$\widetilde{\mathrm{E}}_2^T$	E_2^T	E_3^T	E_3^L	$\mathrm{E}_3^{L'}$	E_4^T	E_4^L	$\mathrm{E}_4^{L'}$	σ
T_{20}		0.2005	0.02917	0.01966	0.02115	0.01656	0.02004	0.02255	0.01758	1.76
T_{40}		123.1	0.1928	0.1305	0.1083	0.0837	0.1320	0.1127	0.0868	2.30
T_{80}	$[-10, 10]$	—	1.520	1.026	0.659	0.534	0.964	0.643	0.519	2.85
T_{160}		—	13.28	8.86	4.74	3.95	8.27	4.65	3.88	3.36
T_{320}		—	159.8	104.8	52.4	45.7	94.9	50.7	44.1	3.50
H_{20}		0.1024	0.02337	0.01716	0.01779	0.01378	0.01639	0.01968	0.01521	1.70
H_{40}		10.37	0.1364	0.1010	0.0871	0.0660	0.1018	0.0897	0.0683	2.07
H_{80}	$[-25, 25]$	—	0.977	0.725	0.484	0.379	0.632	0.494	0.389	2.58
H_{160}		—	6.80	5.44	3.02	2.37	5.19	3.06	2.39	2.87
H_{320}		—	71.7	61.8	31.9	25.9	47.6	31.9	25.1	2.77
M_{21}		0.9342	0.01787	0.009825	0.01176	0.008525	0.009681	0.01172	0.009060	2.10
M_{41}		—	0.1047	0.05708	0.05195	0.03939	0.05636	0.05217	0.04041	2.66
M_{81}	$[-1, 1]$	—	0.7824	0.4081	0.3086	0.2459	0.4023	0.3012	0.2349	3.18
M_{161}		—	6.937	3.707	2.258	1.887	3.630	2.184	1.786	3.68
M_{321}		—	85.82	43.78	25.58	21.94	42.03	24.49	20.65	3.91
S_{100}		1.039	1.180	0.615	0.509	0.404	0.596	0.500	0.393	2.92
S_{200}	$[-10, 10]$	8.019	11.17	5.70	3.87	3.24	5.52	3.72	3.09	3.45
S_{400}		103.4	154.0	76.2	45.8	41.1	73.8	43.6	39.7	3.75
S_{800}		1556	2322	1160	636	589	1123	569	561	3.94

versions of Eval with the same convergence order is mainly due to the inferior recursive Lagrange form with only quadratic convergence that is used for f'. In fact, if we use $\square_3^T$ instead of $\square_2^L$ for the range of f' in E_3^L, then the tree sizes are almost identical to those of E_3^T, and likewise for E_4^L versus E_4^T. However, the price of larger subdivision trees seems to be well compensated for when it comes to the actual performance of the different Eval variants.

Our experimental platform is a Windows 10 laptop with 1.8 GHz Intel Core i7-8565U processor and 16 GB RAM. The average running times (over $1600/n$ runs for T_n, H_n, $800/k$ runs for M_{2k+1}, and $4000/n$ runs for S_n) of our eight versions of Eval on our list of 19 polynomials are obtained by using two kinds of computer arithmetic: 1024-bit floating point arithmetic (Table 3) and multi-precision rational arithmetic (Table 4). No times (and tree sizes in Table 2) are reported, if an Eval version did not terminate within 1 hour. Both arithmetic variants come from the multiple-precision arithmetic library *GMP* [10]. For rational arithmetic, we replaced the constant $\sqrt{3}$ in the definitions of Ω_3, $\square_2^L$, and $\square_2^{L'}$ with the slightly larger rational number $17320508075688773/10^{16}$, so that the validity of the bounds is not altered. Moreover, we temporarily switch to 1024-bit floating point arithmetic for computing square roots. The latter is unavoidable when computing the exact ranges of cubic polynomials (see Sec. A.3) and thus needed by the range functions with quartic convergence.

We draw several conclusions from the tables: 1) The Eval version $\widetilde{\mathrm{E}}_2^T$ based on minimal forms may be utterly non-competitive with the maximal form E_2^T (the former timed out after 1 hour for degrees $n > 40$ for the first 3 sets of polynomials). We expect the same conclusion for other minimal forms. 2) The Eval versions based on recursive Lagrange forms outperform the ones based on Taylor forms with the same convergence order, despite the larger subdivision trees. We attribute this to the fewer number (about one-third) of derivative values that are computed. 3) It does not pay to use range functions with quartic convergence order, because the overhead of computing exact ranges of cubic instead of quadratic polynomials seems to cancel the advantage of smaller tree sizes. 4) Based on speed and implementation simplicity, we declare the Eval variant $\mathrm{E}_3^{L'}$ as the winner in this comparison. 5) Viewing E_2^T as the state-of-art, we see that $\mathrm{E}_3^{L'}$ is at least twice as fast but asymptotically 3 to 4.5 times faster: this is seen in the speedup σ, defined as the ratio of the timings $\mathrm{E}_2^T : \mathrm{E}_3^{L'}$, in the last column of Tables 3 and 4.

6 CONCLUSIONS

Bounding the range of a function is an important problem in many scientific disciplines, but most range functions have only quadratic convergence order. Higher convergence orders and other improvements are particularly important for generic root finding applications (of which root isolation is only one aspect). This is because root finding is a demanding application, in part because its long history and literature has produced some very good algorithms which any new algorithm must contend with. The upshot is that tight and efficient range functions are in demand.

In this paper, we use the framework of Cornelius and Lohner [7] to investigate range functions of any order convergence k. For a fixed k, we explore the two formulations of the remainder form: Taylor expansion and Lagrange interpolation. We see that this remainder form can be refined to any "level" n ($n \geq k$); the remainder form is minimal if $n = k$ and maximal when $n = \infty$. Experimentally, we show that the minimal form may be far inferior to the maximal form. This phenomenon should be investigated theoretically.

We then proceed to a holistic comparison of the resulting recursive Lagrange forms and the generalized Taylor forms with cubic and quartic convergence in the context of the Eval root isolation procedure. Our empirical study suggests that both forms behave similarly and that the recursive Lagrange form with cubic convergence is particularly well-suited for Eval, giving a significant speed-up, compared to the state of the art.

One limitation of our empirical work is that the floating point version of Eval has not accounted for round-off errors. But we verified experimentally that our floating point version agrees with that of the rational arithmetic version in two ways: (a) they generate subdivision trees of the same size (that explains why there is only one Table 2) and (b) they both count the same number of isolator intervals. To address the issues of implementation including errors from rounding in machine arithmetic, it is possible to apply the 3-levels "AIE methodology" in [31] to our algorithms.

ACKNOWLEDGMENTS

Lucas was supported by a UROP Fellowship (Summer 2019) from the Faculty of Informatics at Università della Svizzera italiana (USI). Chee's work began under a USI Visiting Professor Fellowship (Fall 2018); additional support comes from NSF Grants # CCF-1564132 and # CCF-2008768.

REFERENCES

[1] R. Becker, M. Sagraloff, V. Sharma, J. Xu, and C. Yap. 2016. Complexity analysis of root clustering for a complex polynomial. *ISSAC '16*. ACM, New York, 71–78.
[2] R. Becker, M. Sagraloff, V. Sharma, and C. Yap. 2018. A near-optimal subdivision algorithm for complex root isolation based on Pellet test and Newton iteration. *J. Symb. Comput.* 86 (2018), 51–96.
[3] M.A. Burr. 2016. Continuous amortization and extensions: With applications to bisection-based root isolation. *J. Symb. Comput.* 77 (2016), 78–126.
[4] M.A. Burr and F. Krahmer. 2012. SqFreeEVAL: An (almost) optimal real-root isolation algorithm. *J. Symb. Comput.* 47 (2012), 153–166.
[5] M.A. Burr, F. Krahmer, and C. Yap. 2009. *Continuous Amortization: A Non-Probabilistic Adaptive Analysis Technique.* Technical Report 136. ECCC. 22 pages.
[6] Core Library. Since 1999. Software download, source, documentation and links. https://cs.nyu.edu/exact/core_pages/intro.html
[7] H. Cornelius and R. Lohner. 1984. Computing the range of values of real functions with accuracy higher than second order. *Computing* 33 (1984), 331–347.
[8] A. Frommer, B. Lang, and M. Schnurr. 2004. A comparison of the Moore and Miranda existence tests. *Computing* 72 (2004), 349–354.
[9] A. Goldsztejn. 2007. Comparison of the Hansen–Sengupta and the Frommer–Lang–Schnurr existence tests. *Computing* 79 (2007), 53–60.
[10] T. Granlund and GMP Development Team. 2015. *GNU MP 6.0: Multiple Precision Arithmetic Library.* Samurai Media Limited, Hong Kong.
[11] E.R. Hansen and G.W. Walster. 2004. *Global Optimization Using Interval Analysis* (2nd ed.). Marcel Dekker, New York.
[12] R. Imbach, V.Y. Pan, and C. Yap. 2018. Implementation of a near-optimal complex root clustering algorithm. In *Mathematical Software – ICMS 2018.* LNCS, Vol. 10931. Springer, Cham, 235–244.
[13] A. Kobel, F. Rouillier, and M. Sagraloff. 2016. Computing real roots of real polynomials ... and now for real! *ISSAC '16*. ACM, New York, 303–310.
[14] L. Lin and C. Yap. 2011. Adaptive isotopic approximation of nonsingular curves: the parameterizability and nonlocal isotopy approach. *Discrete Comput. Geom.* 45 (2011), 760–795.
[15] L. Lin, C. Yap, and J. Yu. 2013. Non-local isotopic approximation of nonsingular surfaces. *Comput.-Aided Des.* 45 (2013), 451–462.
[16] J.M. McNamee. 2007. *Numerical Methods for Roots of Polynomials, Part 1.* Elsevier, Amsterdam.
[17] J.M. McNamee and V.Y. Pan. 2013. *Numerical Methods for Roots of Polynomials, Part 2.* Elsevier, Amsterdam.
[18] D.P. Mitchell. 1990. Robust ray intersection with interval arithmetic. In *Proc. Graph. Interface (GI '90).* Canadian Info. Processing Soc., Toronto, 68–74.
[19] R.E. Moore, R.B. Kearfott, and M.J. Cloud. 2009. *Introduction to Interval Analysis.* Society for Industrial and Applied Mathematics, Philadelphia.
[20] A. Neumaier. 1990. *Interval Methods for Systems of Equations.* Cambridge University Press, Cambridge.
[21] V.Y. Pan. 2002. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *J. Symb. Comput.* 33 (2002), 701–733.
[22] A. Plantinga and G. Vegter. 2004. Isotopic approximation of implicit curves and surfaces. In *Proc. Symp. Geom. Process. (SGP '04).* ACM, New York, 245–254.
[23] H. Ratschek and J. Rokne. 1984. *Computer Methods for the Range of Functions.* Ellis Horwood Limited, Chichester.
[24] M. Sagraloff and K. Mehlhorn. 2016. Computing real roots of real polynomials. *J. Symb. Comput.* 73 (2016), 46–86.
[25] M. Sagraloff and C.K. Yap. 2011. A simple but exact and efficient algorithm for complex root isolation. *ISSAC '11*. ACM, New York, 353–360.
[26] A. Schönhage. 1982. The Fundamental Theorem of Algebra in Terms of Computational Complexity. www.informatik.uni-bonn.de/~schoe/fdthmrep.ps.gz.
[27] A. Shadrin. 1995. Error bounds for Lagrange interpolation. *J. Approx. Theory* 80 (1995), 25–49.
[28] V. Sharma and C.K. Yap. 2012. Near optimal tree size bounds on a simple real root isolation algorithm. *ISSAC '12*. ACM, New York, 319–326.
[29] V. Stahl. 1995. *Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations.* Ph.D. Thesis. Johannes Kepler Univ., Linz.
[30] J. von zur Gathen and J. Gerhard. 1997. Fast algorithms for Taylor shifts and certain difference equations. *ISSAC '97*. ACM, New York, 40–47.
[31] J. Xu and C. Yap. 2019. Effective subdivision algorithm for isolating zeros of real systems of equations, with complexity analysis. *ISSAC '19*. ACM, NY, 355–362.
[32] C. Yap, M. Sagraloff, and V. Sharma. 2013. Analytic root clustering: A complete algorithm using soft zero tests. In *The Nature of Computation. Logic, Algorithms, Applications.* LNCS, Vol. 7921. Springer, Berlin, 434–444.
[33] J. Yu, C. Yap, Z. Du, S. Pion, and H. Brönnimann. 2010. The design of Core 2: A library for exact numeric computation in geometry and algebra. In *Mathematical Software – ICMS 2010.* LNCS, Vol. 6327. Springer, Berlin, 121–141.

A EXACT RANGES FOR LOW DEGREE POLYNOMIALS

Let $I = [a, b]$, $r = (b - a)/2$, $m = (a + b)/2$, and

$$A := \min\{g(a), g(b)\}, \qquad B := \max\{g(a), g(b)\}. \tag{30}$$

A.1 Linear polynomials

The exact range $g(I) = [A, B]$ of the linear polynomial $g_1(x) = c_0 + c_1(x - m)$ is given by the assignment in (30), because g is monotonic. The range can also be expressed as

$$g(I) = c_0 + r[-1, 1]c_1.$$

A.2 Quadratic polynomials

To determine the exact range $g(I) = [A, B]$ of the quadratic polynomial $g(x) = c_0 + c_1(x - m) + c_2(x - m)^2$ we first observe that the extremum of g occurs at

$$x^* = m - \frac{c_1}{2c_2},$$

which is inside I, if and only if $|c_1| < 2|c_2|r$. If $x^* \notin I$, then g is monotonic on I and the assignment in (30) gives the correct range. Otherwise, we check the sign of c_2 to see if g has a minimum ($c_2 > 0$) or a maximum ($c_2 < 0$) at x^* and accordingly replace A or B with

$$g(x^*) = c_0 - \frac{c_1^2}{4c_2}.$$

Note that the special case of g being linear (or constant) with $c_2 = 0$ is automatically handled correctly by this procedure.

A.3 Cubic polynomials

To find the exact range $g(I) = [A, B]$ of the cubic polynomial $g(x) = c_0 + c_1(x - m) + c_2(x - m)^2 + c_3(x - m)^3$, we assume that $c_3 \neq 0$. If $c_3 = 0$, then $g(x)$ is a polynomial of degree at most two and its range can be found with the method in Sec. A.2.

We now analyze the stationary points of g by considering the quadratic equation

$$g'(x) = c_1 + 2c_2(x - m) + 3c_3(x - m)^2 = 0$$

and in particular its discriminant

$$\Delta = c_2^2 - 3c_1c_3.$$

If $\Delta < 0$, then g does not have any local extrema, and if $\Delta = 0$, then g has a stationary point of inflection. In both cases, g is monotonic and its range is given by the assignment in (30).

If $\Delta > 0$, then g has a local minimum at x^- and a local maximum at x^+, where

$$x^\pm = m - \frac{c_2 \pm \sqrt{\Delta}}{3c_3}. \tag{31}$$

To determine the range of g, we need to know whether x^- and x^+ are inside or outside I and must distinguish four cases:

1) If $x^-, x^+ \in (a, b)$, then $g(I) = [\min\{A, g(x^-)\}, \max\{B, g(x^+)\}]$.
2) If $x^- \in (a, b)$ and $x^+ \notin (a, b)$, then $g(I) = [g(x^-), B]$.
3) If $x^+ \in (a, b)$ and $x^- \notin (a, b)$, then $g(I) = [A, g(x^+)]$.
4) If $x^-, x^+ \notin (a, b)$, then g is monotonic over I and $g(I) = [A, B]$.

Algorithm 2 Computing the exact range $g(I) = [A, B]$ of the cubic polynomial $g(x) = \sum_{i=0}^{3} c_i(x - m)^i$ with $c_3 \neq 0$ over $I = [a, b]$.

```
A := min{g(a), g(b)}
B := max{g(a), g(b)}
Δ := c_2^2 − 3c_1c_3
if Δ > 0 then
    L := sgn(c_3)(c_1 + 3c_3r^2)
    R := 2|c_2|r
    if L > R then
        if |c_2| < 3|c_3|r then
            x^- := m − (c_2 − √Δ)/(3c_3)
            x^+ := m − (c_2 + √Δ)/(3c_3)
            A := min{A, g(x^-)}
            B := max{B, g(x^+)}
    else if L > −R then
        if c_2 > 0 then
            x^- := m − (c_2 − √Δ)/(3c_3)
            A := g(x^-)
        else if c_2 < 0 then
            x^+ := m − (c_2 + √Δ)/(3c_3)
            B := g(x^+)
return [A, B]
```

In all four cases, A and B are assumed to be as in (30).

We shall now work out rather simple conditions for detecting these cases. To this end, we first conclude from (31) that

$$x^\pm \in (a, b) \quad \Leftrightarrow \quad |x^\pm - m| < r \quad \Leftrightarrow \quad |c_2 \pm \sqrt{\Delta}| < 3|c_3|r.$$

Since the larger of the two values $|c_2 - \sqrt{\Delta}|$ and $|c_2 + \sqrt{\Delta}|$ equals $|c_2| + \sqrt{\Delta}$, it is clear that both x^- and x^+ are inside I, if and only if

$$|c_2| + \sqrt{\Delta} < 3|c_3|r \quad \Leftrightarrow \quad \sqrt{\Delta} < 3|c_3|r - |c_2|. \tag{32}$$

Squaring both sides of the last inequality gives

$$\Delta = c_2^2 - 3c_1c_3 < 9c_3^2r^2 - 6|c_2||c_3|r + c_2^2$$
$$\Leftrightarrow \quad 2|c_2|r < \frac{c_3}{|c_3|}(c_1 + 3c_3r^2),$$

but, of course, this is equivalent to (32) only if the right-hand side of the last inequality is positive. Hence, the condition for the first case above is

$$x^-, x^+ \in (a, b)$$
$$\Leftrightarrow \quad \sigma(c_1 + 3c_3r^2) > 2|c_2|r \quad \text{and} \quad |c_2| < 3|c_3|r, \tag{33}$$

where $\sigma = c_3/|c_3| = \operatorname{sgn}(c_3)$. In a similar way, it can be shown that

$$x^- \in (a, b), \quad x^+ \notin (a, b)$$
$$\Leftrightarrow \quad 2|c_2|r \geq \sigma(c_1 + 3c_3r^2) > -2|c_2|r \quad \text{and} \quad c_2 > 0$$

and

$$x^+ \in (a, b), \quad x^- \notin (a, b)$$
$$\Leftrightarrow \quad 2|c_2|r \geq \sigma(c_1 + 3c_3r^2) > -2|c_2|r \quad \text{and} \quad c_2 < 0,$$

and the condition for the last case above is simply that none of these three cases occur. Note that this includes the case when $c_2 = 0$ and $c_1 + 3c_3r^2 = 0$, which turns out to be equivalent to the condition $\{x^-, x^+\} = \{a, b\}$. Overall, this analysis leads to Algo. 2.