Quadratic Interval Refinement for Real Roots

John Abbott
Dipartimento di Matematica
Università di Genova
Genova 16146, Italy
abbott@dima.unige.it

Abstract

We present a new algorithm for refining a real interval containing a single real root: the new method combines the robustness of the classical Bisection algorithm with the speed of the Newton-Raphson method; that is, our method exhibits quadratic convergence when refining isolating intervals of simple roots of polynomials (and other well-behaved functions). We assume the use of arbitrary precision rational arithmetic. Unlike Newton-Raphson our method does not need to evaluate the derivative.

1 Introduction

Note This is a much delayed presentation of the QIR algorithm which was developed in 2002, its software implementation was released publicly in 2003 (as part of CoCoA 4.3, and demonstrated at MEGA 2003), and the details were originally presented as a poster at the ISSAC 2006 conference. As a consequence, this article contains some *temporal forward references*.

The task of computing approximations to the real roots of functions has attracted much interest ever since antiquity, and many different methods have been devised — for instance, see the Wikipedia entry *Root-finding algorithm* for a very brief survey. The methods fall into two categories: those designed to work with limited precision arithmetic, and those which presuppose no limit. Our new method belongs to the latter category.

Typically the process of approximating the real roots of a univariate polynomial comprises two phases: *root isolation* where the distinct roots are separated into disjoint intervals, followed by *root refinement* where the approximations of the roots are improved until they are within specified limits. This paper assumes that isolation has already taken place, and concerns itself with the refinement of a single interval.

We recall two particularly well-known methods for root refinement: *Newton-Raphson* and *Bisection*. Newton-Raphson dates back over three hundred years, and is simple and fast (under suitable starting conditions). Bisection is slower than Newton-Raphson but more robust (with much simpler starting conditions guaranteeing success). Our new technique uses linear interpolation (as in *regula falsi*) together with a simple, adaptive process for reducing the interval width; it combines the robustness of Bisection with the rapid convergence of Newton-Raphson (for simple roots of nice

fnctions). The algorithm is implemented as an integral part of CoCoA (from version 4.3 onwards). We call the new algorithm **Quadratic Interval Refinement**, or simply **QIR**.

Prerequisites for using QIR are knowledge of an initial interval I for which the function has opposite signs at its two (rational) end points, and a procedure which evaluates f at a given rational point with arbitrary precision. Naturally, the function f should be continuous on I, but we do not require that f be differentiable. It is best if the interval I contains a single, simple root of f; furthermore, if f has a valid Taylor expansion about that single, simple root then convergence of QIR will ultimately be quadratic, like Newton-Raphson.

In fact, like Bisection, QIR will work correctly even with discontinuous functions provided that there is no discontinuity causing a sign change. Also like Bisection, if I contains several roots then QIR will eventually discard all but one of them as refinement proceeds.

1.1 Earlier Similar Work

Collins and Krandick presented in [6] two algorithms and a heuristic all based on Newton-Raphson. They introduced the term *binary rational* to mean a rational of the form $n/2^k$ for some $n \in \mathbb{Z}$. The motivation for this definition is the speed of computation with numbers of the form 2^k; they exploited this speed in their `BINARY-RATIONAL-NEWTON` algorithm. The same reasoning led to our use of powers of 4 for the refinement factor in QIR; in fact, if the end points of the initial interval are binary rationals then the end points of the interval output by QIR will also be binary rationals. Despite this common use of binary rationals, the two algorithms refine the interval in each iteration in quite different ways; in particular `BINARY-RATIONAL-NEWTON` requires computing the derivative, and also imposes stricter conditions on the starting interval.

Researchers in the realm of validated floating-point computation have also studied techniques for interval root approximation. A version of Newton-Raphson for interval arithmetic was apparently first published by Moore in 1966 (see §8.1 in [9]). The algorithm is simple and achieves quadratic convergence (for simple roots of rational functions); like classical Newton-Raphson it computes values of the derivative, and also requires that the derivative (considered as an interval function) does not vanish on the initial isolating interval. It presupposes the availability of an underlying system of basic interval arithmetic, but has no other special requirements.

1.2 Subsequent Developments

The simplicity and efficiency of QIR has attracted considerable interest from other researchers. Here is a brief summary of some of the developments made so far.

An interesting and practical generalization of our algorithm to the case of approximate arithmetic was given in [7], together with a detailed bit-complexity analysis. They called their method AQIR, and reported a significant improvement in speed over the exact arithmetic version described here. Importantly, AQIR relaxes QIR's requirement for an exact evaluation "black box" for f, and works instead with a "black box" which computes the value to within a specified accuracy.

Another remarkable development of the idea underlying QIR was presented in [10] where a new, efficient algorithm for real root isolation was devised. Combining this new isolation method with AQIR produces a simple, complete algorithm for approximating the real roots of a polynomial $f \in \mathbb{Q}[x]$ with nearly optimal bit complexity [11].

1.3 Notation

For a real number x we write round(x) to mean the integer closest to x; either candidate may be chosen in the case of a tie. We write $I(a,b)$ to denote the open interval $\{x \in \mathbb{R} : a < x < b\}$; and we denote its width by $|I(a,b)| = b - a$.

We abuse terminology harmlessly by taking **isolating interval** to mean an interval $I(a,b)$ for which $f(a)$ and $f(b)$ have opposite signs — we do not require that the interval contain just a single root of f, though if there are several, QIR may discard all except one as it refines.

By **Newton-Raphson** we refer to the root approximation method starting from some given x_0 and generating successive iterates using the formula $x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1})$.

By **Bisection** we refer to the interval refinement method where the interval $I(a,b)$ is refined to either $I\big(a, \frac{1}{2}(a+b)\big)$ or $I\big(\frac{1}{2}(a+b), b\big)$, the choice depending on the sign of $f\big(\frac{1}{2}(a+b)\big)$.

2 Motivation

Why invent a new method when Newton-Raphson works so well? QIR offers a simple unified approach which can refine any isolating interval with a rate of convergence and a computational cost comparable to that of Newton-Raphson. Moreover, QIR has several advantages over Newton-Raphson:

(a) the criteria guaranteeing convergence are simple and easy to verify;

(b) there is no need to evaluate the derivative of the function whose zeroes we are approximating;

(c) the successive approximants have simple denominators;

(d) roots are approximated by intervals containing them, so the accuracy is quite explicit.

A significant difficulty with Newton-Raphson is obtaining a suitable starting value given an isolating interval: this is the essence of point (a). It may be necessary to use some other interval refinement method prior to using Newton-Raphson (*e.g.* to be sure that the derivative does not vanish in the interval). Point (b) poses no problem when approximating roots of explicit polynomials since the derivative can readily be evaluated, but in other cases it may not be easy to obtain values of the derivative. Point (c) is relevant only when using arbitrary precision rational arithmetic, as is the case with CoCoA. Point (d) is important when a guarantee of the accuracy obtained is desired. With some cunning (*e.g.* see [6]) one can alleviate weaknesses (c) and (d) of Newton-Raphson: *e.g.* replacing approximants by similar values having simpler denominators of about the right size, and using the derivative to estimate the width of an interval containing the root.

Example QIR can never fail, whereas Newton-Raphson can. The polynomial $f(x) = x^3 - 2x + 2$ has a single real root $\alpha \approx -1.77$. QIR will happily refine any starting interval containing the root, but Newton-Raphson fails to converge when given a starting value x_0 in the range $-0.12 < x_0 < 0.13$ which is not so far from the root!

As also noted in [6], one should avoid naive use of Newton-Raphson with exact rational arithmetic because the sizes of the numerators and denominators of successive approximants can increase rapidly: typically the k-th approximant will have numerator (and also denominator) containing $O(d^k)$ digits where d is the degree of the polynomial whose root is being sought.

Example The polynomial $x^5 - 2$ contains a real root in the interval $I(1, 2)$. Using rational arithmetic to estimate the root with an error less than 2^{-32} starting from the initial approximation $x_0 = 1$ requires five Newton-Raphson iterations, and produces an approximant having a denominator with over 500 digits. Alternatively starting from $x_0 = 2$ requires seven iterations, and the final approximant has a denominator with over 17000 digits. In contrast, QIR obtains an interval of width $\leq 2^{-32}$ after six iterations, and no value in the computation has more than 50 digits.

3 Description of the Method

QIR works by repeatedly narrowing the isolating interval until the width is smaller than the prescribed limit. Each individual narrowing step receives an **initial interval** and a **refinement factor** $N \in \mathbb{N}$ by which it must reduce the interval width; it uses *discretized linear interpolation* to guess a narrower interval containing the root. If the guess was good, the interval is updated; if not, the interval is unchanged, and the narrowing step returns an indication of failure. The refinement factor is increased after every successful narrowing step.

A *narrowing step* conceptually divides its input interval into N consecutive equal-width sub-intervals. It then uses linear interpolation to locate a sub-interval in which it guesses the root to lie. The guess is tested by evaluation at the end points of the chosen sub-interval. If the guess was good then the initial interval is replaced by the chosen sub-interval; otherwise an indication of failure is returned. The case $N = 4$ is handled specially: the initial interval is *always* replaced by the correct sub-interval, *but* failure is indicated if linear interpolation led to a bad guess.

The refinement factor is varied according to the following rule:

- after a successful narrowing step, $N \leftarrow N^2$
- after a failed narrowing step, if $N > 4$ then $N \leftarrow \sqrt{N}$

This strategy is inspired by the knowledge that linear interpolation produces approximations whose errors decrease roughly quadratically when sufficiently close to a root of a well-behaved function.

Obviously, if one of the evaluation points happens to be the exact root ξ then this value is returned as an exact root. This can occur only if ξ is rational and with "suitable" denominator.

3.1 An Illustrative Example

The behaviour of QIR prior to the onset of quadratic convergence can be quite complex. To illustrate this we present a simple example where QIR experiences a mixture of successes and failures before true quadratic convergence begins.

Given the polynomial $f = 1111\,x^2 - 1$ the root isolator in CoCoA produces the interval $I(0, 2)$ for the positive root. Starting from this interval QIR initially enjoys two **success**es, a **failure**, another **success** and another **failure**; at this point the refinement factor is 16 and quadratic convergence begins. Proceeding we find that after a total of 13 iterations in `RefineInterval` we have an interval of width $2^{-518} \approx 10^{-156}$; and just 3 more iterations refines this to a width of $2^{-4102} \approx 10^{-1234}$. As each iteration makes 2 evaluations, we have needed only 32 evaluations to obtain the root to over 1000 digits!

4 The Algorithm

Here we present explicitly the algorithm using a pseudo-language. The actual source code is contained in the CoCoA package `RealRoots.cpkg5` and is publicly available as part of the standard distribution of CoCoA (from version 4.3 onwards) — see the web site [2].

4.1 Main routine: `RefineInterval`

INPUT: f the function whose zero is being sought,
$I(x_{lo}, x_{hi})$ an open interval with rational end points $x_{lo} < x_{hi}$, and for which $f(x_{lo}) \cdot f(x_{hi}) < 0$
$\varepsilon > 0$ an upper bound for the width of the refined interval.

OUTPUT: $I(\xi_{lo}, \xi_{hi})$ an open sub-interval of width at most ε having rational end points and such that $f(\xi_{lo}) \cdot f(\xi_{hi}) < 0$. Exceptionally the output may be the exact value of a root ξ.

(1) Initialize refinement factor $N \leftarrow 4$, and interval $I \leftarrow I(x_{lo}, x_{hi})$.

(2) While $|I| > \varepsilon$ do steps (2.1)–(2.4).

(2.1) Apply `RefineIntervalByFactor` to f, I and N — this usually modifies the interval I.

(2.2) If an **exact root** ξ has been found, simply return ξ.

(2.3) If **success** was reported then replace $N \leftarrow N^2$;
otherwise (**failure** was reported) replace $N \leftarrow \sqrt{N}$ if $N > 4$.

(3) Return the interval I.

Remark This pseudo-code reflects the current CoCoA implementation; we mention here some variants for the case when **failure** occurs. For an interval containing a single, simple root of a nice function, **failure** can happen only "a few times" (see the proof in section 4.5) so it matters little which variant is employed.

- An alternative strategy for reducing N in step (2.3) would be simply to set it to 4. Compared to our step-by-step reduction this strategy would avoid a possible succession of failing steps where N is repeatedly reduced until **success** comes or N reaches 4 (in the worst case).
- When **failure** occurs with $N > 4$ we have a "wasted" iteration: two evaluations but no refinement. In fact, we could use the bad sub-interval to let us perform some refinement since we know there is a sign change outside the sub-interval (and also on which side of it). Alternatively, as a referee suggested, we could perform explicitly a bisection step.

4.2 Auxiliary procedure: `RefineIntervalByFactor` when $N > 4$

INPUTS: f the function whose zero we are seeking,
$I = I(x_{lo}, x_{hi})$ an open interval with rational end points $x_{lo} < x_{hi}$ such that $f(x_{lo}) \cdot f(x_{hi}) < 0$
$N \in \mathbb{N}$ the refinement factor — here we assume $N > 4$, see §4.3 below for the case $N = 4$.

OUTPUT: **success** or **failure**; if **success** then the interval I is replaced by an open sub-interval $I(\xi_{lo}, \xi_{hi})$ having rational end points, whose width is $\frac{1}{N}|I|$, and such that $f(\xi_{lo}) \cdot f(\xi_{hi}) < 0$. Exceptionally, the output may be an **exact root** ξ.

(1) Conceptually divide the interval I into N consecutive sub-intervals of equal width, $w = \frac{1}{N}|I|$. Define $x_0, x_1, \ldots, x_N$ the end points of these sub-intervals: $x_0 = x_{lo}$, and $x_k = x_{k-1} + w$.

(2) Using linear interpolation predict approximate position of the zero: *i.e.* determine the closest end point $\hat{x} = x_\kappa$ where $\kappa = \text{round}(N \cdot f(x_{lo})/(f(x_{lo}) - f(x_{hi})))$

(3) **Evaluate** $f(\hat{x})$. If $f(\hat{x}) = 0$ then return **exact root** $\hat{x}$.

(4) *Check whether our prediction was good: does f change sign in one of the sub-intervals having $\hat{x}$ as an end point?*
If $f(\hat{x})$ has the same sign as $f(x_{lo})$, *check the interval to the right:*

(4.1) **Evaluate** $f(\hat{x} + w)$
(4.2) If $f(\hat{x} + w) = 0$ then return **exact root** $\hat{x} + w$.
(4.3) If $f(\hat{x} + w)$ has the same sign as $f(\hat{x})$ then return **failure**.
(4.4) Otherwise replace $I \leftarrow I(\hat{x}, \hat{x} + w)$ and return **success**.

Otherwise $f(\hat{x})$ *has the same sign as $f(x_{hi})$, so check the interval to the left:*

(4.5) **Evaluate** $f(\hat{x} - w)$
(4.6) If $f(\hat{x} - w) = 0$ then return **exact root** $\hat{x} - w$.
(4.7) If $f(\hat{x} - w)$ has the same sign as $f(\hat{x})$ then return **failure**,
(4.8) Otherwise replace $I \leftarrow I(\hat{x} - w, \hat{x})$ and return **success**.

Remark For efficiency, it is best to include also $f(x_{lo})$ and $f(x_{hi})$ as input parameters.

4.3 Auxiliary procedure: `RefineIntervalByFactor` when $N = 4$

This is very similar to the general case; the important difference is that the input interval I is *always* replaced by a smaller one (of width $\frac{1}{4}|I|$).

INPUTS: same as general case, except we know $N = 4$

OUTPUT: **success** or **failure**; in either case the interval I is replaced by an open sub-interval $I(\xi_{lo}, \xi_{hi})$ having rational end points, whose width is $\frac{1}{4}|I|$, and such that $f(\xi_{lo}) \cdot f(\xi_{hi}) < 0$. Exceptionally, the output may be an **exact root** ξ.

(1) Conceptually divide the interval I into 4 consecutive sub-intervals of equal width: $w = \frac{1}{4}|I|$. Define the end points $x_0, \ldots, x_4$ as in the general case.

(2) Using linear interpolation predict approximate position of the zero: *i.e.* determine the **left end point** $\hat{x} = x_\kappa$ where $\kappa = \lfloor N \cdot f(x_{lo})/(f(x_{lo}) - f(x_{hi})) \rfloor$

(3) Reduce the width of I by performing **two bisection steps**; exceptionally this may return an **exact root**.

(4) If the left end point of I equals x_κ return **success**, otherwise return **failure**.

Remark We handle the base case of $N = 4$ this way because it guarantees refinement by a factor of 4 at a cost of just 2 evaluations, and it returns **success** only when linear interpolation has given a fairly good estimate of the root position (and so it makes sense to try a larger refinement factor).

Remark In [8] they presented the EQIR algorithm, which is identical to QIR except for the handling of small N: their base case is when $N = 2$, it performs a bisection step, and is always **success**ful, so N is always increased to 4 for the next iteration; they treat the case $N = 4$ the same as for general, larger values. However, in some cases EQIR needs more evaluations than QIR during the initial phase: for instance, let $f(x) = 3\,x^{100} - 1$ and take the initial interval $I(0, 1)$. QIR starts with refinement factor 4 and obtains the interval $I\left(\frac{253}{256}, \frac{254}{256}\right)$ after just 10 evaluations. In contrast, EQIR starting with refinement factor 4 requires 22 evaluations to find the same interval. Of course, if high precision is sought and refinement goes on to make several truly quadratic iterations then these initial differences become largely insignificant.

4.4 Termination of the main algorithm

We show that the algorithm `RefineInterval` always terminates. In the loop, if $N = 4$ then the interval is always narrowed (by a factor of 4); if $N > 4$ then either the interval is narrowed (by more than a factor of 4), or N is reduced. Since the loop terminates when the interval has width less than ε, and since each narrowing reduces the width by at least a factor of 4, there can be *only a finite number of narrowings*. If no narrowing occurs (*i.e.* $N > 4$ and **failure** was reported) then N is reduced to its own square-root, but as N is never smaller than 4, there can be *only finitely many iterations in which the interval is not narrowed*.

4.5 Quadratic Convergence

The mathematical justification underlying the convergence rate of QIR is very similar to that underlying Newton-Raphson. We assume that the isolating interval I contains a single, simple root ξ and that the function f admits a Taylor expansion centred on ξ valid in an open neighbourhood containing I: *i.e.* whenever $\xi + \delta \in I$ we have for some exponent $k \geq 2$

$$f(\xi + \delta) = C_1 \cdot \delta + C_2 \cdot \delta^k + O(\delta^{k+1})$$

with suitable non-zero constants C_1 and C_2; we know that $C_1 \neq 0$ since ξ is a simple root. We are excluding the trivial case where f is linear on I.

Let x_{lo} and x_{hi} be the two end points of the interval I and let its width be $\varepsilon = x_{hi} - x_{lo}$. From the Taylor expansion we have

$$\begin{aligned} f(x_{lo}) &= C_1 \cdot (x_{lo} - \xi) + C_2 \cdot (x_{lo} - \xi)^k + O(x_{lo} - \xi)^{k+1} \\ f(x_{hi}) &= C_1 \cdot (x_{hi} - \xi) + C_2 \cdot (x_{hi} - \xi)^k + O(x_{hi} - \xi)^{k+1} \end{aligned}$$

Estimating ξ by linear interpolation gives $\hat{\xi} = x_{lo} + \lambda \cdot (x_{hi} - x_{lo})$ where $\lambda = f(x_{lo})/(f(x_{lo}) - f(x_{hi}))$. Observe that $f(x_{lo}) - f(x_{hi}) = C_1\varepsilon + C_2\eta + O(\varepsilon^{k+1})$ where $|\eta| = \left|(x_{lo} - \xi)^k - (x_{hi} - \xi)^k\right| \leq \varepsilon^k$. Substituting and simplifying gives $\hat{\xi} = x_{lo} + (\xi - x_{lo}) - \frac{C_2}{C_1}(x_{lo} - \xi)(\frac{\eta}{\varepsilon} - (x_{lo} - \xi)^{k-1}) + O(\varepsilon^{k+1})$. Rearranging gives $|\hat{\xi} - \xi| < \left|\frac{C_2}{C_1}(x_{lo} - \xi)(\frac{\eta}{\varepsilon} - (x_{lo} - \xi)^{k-1})\right| + |O(\varepsilon^{k+1})|$. For sufficiently small ε we may assume the contribution $|O(\varepsilon^{k+1})| < \left|\varepsilon^2 C_2/C_1\right|$, and thus conclude that $|\hat{\xi} - \xi| < 3\varepsilon^2 C_2/C_1$.

Now we show that under these conditions convergence of QIR is eventually quadratic. We say that an isolating interval I of width ε is **narrow enough** for f whenever $\varepsilon\, C_2/C_1 < \frac{1}{12}$. Observe that, starting from any isolating interval, this condition will be satisfied after only finitely many iterations. Henceforth we shall assume I is *narrow enough*.

When applying the algorithm `RefineIntervalByFactor` to the parameters (f, I, N) we shall say that N is **small enough for** I if $N \cdot \varepsilon\, C_2/C_1 < \frac{1}{3}$. Note that $N = 4$ is always *small enough* because we have assumed that I is *narrow enough*.

Suppose that algorithm `RefineIntervalByFactor` is called on (f, I, N) where N is *small enough* for I. The condition $\varepsilon\, C_2/C_1 < \frac{1}{12}$ implies that linear interpolation gives an estimate $\hat{\xi}$ whose distance from ξ does not exceed $3\varepsilon^2 C_2/C_1 < \varepsilon/N$. Hence step (2) of `RefineIntervalByFactor` makes a good prediction, and so **success** will be returned. Denote the narrowed interval by I'. Now, the next iteration of `RefineInterval` will call `RefineIntervalByFactor` on the parameters (f, I', N^2). Since I' is narrower than I it is surely *narrow enough* for f. Moreover, one may easily verify that N^2 is *small enough* for I'. Hence, by induction, all subsequent calls to `RefineIntervalByFactor` will return **success**, and convergence is therefore quadratic.

In contrast, while still assuming that I is *narrow enough* for f, if `RefineIntervalByFactor` is called when N is not *small enough* for I then **failure** may occur. Since N is not *small enough*, we must have $N > 4$. So when **failure** occurs the refinement factor N is reduced. Hence, there can be only finitely many **failure**s before N becomes *small enough* for I, and then guaranteed quadratic convergence ensues.

4.6 Weaknesses

A weakness of QIR as described here is the cost of evaluating f exactly at various rational points. For instance, if f is a high degree polynomial and the evaluation point x is a rational with large denominator then $f(x)$ is likely to have an enormous denominator, being roughly the d-th power of the denominator of x where d is the degree of f. A mitigating factor in the CoCoA implementation is that the denominator is always a power of 2, allowing faster computation. Nevertheless, this matter was subsequently thoroughly investigated in [7], and effectively resolved by their AQIR algorithm, the approximate variant of QIR.

As presented here QIR may calculate an interval rather narrower than the specified limit. This is a "minor defect" because it simply means that the last two evaluations of f will be at rational numbers with needlessly large denominators (with up to twice as many digits as truly required), and thus will cost more than is strictly necessary. In practice, one could just reduce the value of the refinement factor N in the final iteration of `RefineInterval` to avoid the overshoot.

5 Experimental Results

We present here a small selection of experimental results. The computer used was a MacBook Pro with 2.4GHz processor "Intel Core 2 duo". The software used was CoCoA 5.0.4 which relies upon the excellent library GMP (version 5.1.3, see [3]) for arbitrary precision integer/rational arithmetic. All timings are measured in seconds; note that CoCoA is single-threaded.

CoCoA includes functions (`RealRoots` and `RealRootRefine`) for approximating the real roots of a (univariate) polynomial with rational coefficients; the function `RealRootRefine` implements QIR, while `RealRoots` is a simplistic implementation of the root isolation method described in [1],

which is based on *Descartes's Rule of Signs*. For clarity, all timings reported below *exclude the cost of obtaining isolating intervals*.

We describe briefly the polynomials used for the timing tests. The first two polynomials are adapted from the FRISCO test suite (see [4] and [5]) since CoCoA cannot represent complex numbers. Both polynomials are designed to be challenging for a root isolator, and indeed the real root isolator implemented in CoCoA fails with *"Recursion too deep"*. However $I(0,1)$ is a valid isolating interval for both polynomials, and it was used as input to QIR. The first polynomial is

$$f_1 = ((c^2x^2 - 3)^4 + c^4x^{18})(c^2x^2 - 3) \qquad \text{for } c = 10^{1000}$$

which has four complex roots quite close to the real positive root, and this delays the onset of quadratic convergence. The second polynomial is

$$f_2 = x^{50} + (10^{50}x - 1)^3$$

which has two complex roots very close to the real positive root, thus delaying the onset of quadratic convergence: note how quickly 8000 digits are obtained compared to the time needed for 1000 digits.

The third and fourth polynomials are interesting because they both have a root which is "almost integer": f_3 has a root close to 45, while f_4 has a root close to 10. Additionally, all their roots are real by construction.

$$f_3 = \text{minimal polynomial of } \sqrt{176} + \sqrt{190} + \sqrt{195} + \sqrt{398} + \sqrt{1482}$$
$$f_4 = \text{minimal polynomial of } \sqrt{99} + \sqrt{627} + \sqrt{661} + \sqrt{778} + \sqrt{929} + \sqrt{1366} + \sqrt{1992}$$

In the table below, the column headed "#roots" indicates how many intervals were refined, and the other columns give the times taken to refine all the intervals to width less than ε.

Poly	#roots	$\varepsilon = 10^{-1000}$	$\varepsilon = 10^{-2000}$	$\varepsilon = 10^{-4000}$	$\varepsilon = 10^{-8000}$
f_1	1	0.04	10	67	89
f_2	1	8.4	8.5	8.7	8.9
f_3	32	1.0	2.3	5.6	13
f_4	128	25	60	138	320

In [12] they use Chebyshev polynomials of the first kind to benchmark their implementation. Here we give some timings using QIR; typically QIR is 4 to 10 times faster than Liang's LZ2, but it is unclear how the processors compare. The timings are for refining a single root. The column *Req. precision* gives the requested maximum interval width; the column *Intvl width* gives the approximate width of the returned interval.

Poly	Req. precision	Time	Intvl width
T_{100}	10^{-1000}	0.09	10^{-1233}
T_{200}	10^{-1000}	0.22	10^{-1234}
T_{400}	10^{-1000}	0.54	10^{-1234}
T_{800}	10^{-1000}	1.3	10^{-1234}
T_{1600}	10^{-1000}	3.2	10^{-1235}
T_{1000}	10^{-500}	0.56	10^{-620}
T_{1000}	10^{-1000}	1.3	10^{-1237}
T_{1000}	10^{-2000}	3.1	10^{-2470}

6 Conclusions

QIR is a simple algorithm which combines robustness and speed without imposing particular conditions on the initial interval beyond the opposing signs of the function at the end points. It is easy to implement, and the open-source implementation supplied with CoCoA demonstrates its genuine competitiveness with the LZ2 method in [12].

While it is clear that the running time of QIR is dominated by the cost of the last evaluations confirming the validity of the returned interval, a proper thorough complexity analysis of QIR (and also AQIR) was produced by Kerber and Sagraloff in [8].

QIR is well suited to the context of exact symbolic computation, whereas Kerber and Sagraloff's subsequent AQIR variant presented in [7] is applicable also to circumstances where only approximate evaluation is possible.

References

[1] F Roullier, P Zimmermann *Efficient Isolation of Polynomial Real Roots* INRIA Rapport de Recherche 4113, Feb 2001.

[2] The CoCoA web site `http://cocoa.dima.unige.it/`

[3] The GMP web site `http://www.swox.com/gmp/`

[4] FRISCO Polynomial Test Suite `http://www-sop.inria.fr/saga/POL/`

[5] MPSolve Benchmarks `http://www.dm.unipi.it/cluster-pages/mpsolve/bench.htm`

[6] G E Collins & W Krandick *A hybrid method for high precision calculation of polynomial real roots* Proc. ISSAC 1993, pp. 47–52

[7] M Kerber & M Sagraloff *Efficient Real Root Approximation*, Proc. ISSAC 2011, pp. 209–216

[8] M Kerber & M Sagraloff *Root Refinement for Real Polynomials* `arXiv:1104.1362` (2011)

[9] R E Moore, R B Kearfott, M J Cloud *Introduction to Interval Analysis* SIAM 2009

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

[11] M Sagraloff & K Mehlhorn *Computing Real Roots of Real Polynomials: An Efficient Method Based on Descartes' Rule of Signs and Newton Iteration* `arXiv:1308.4088v1` (2013)

[12] Y Liang *Real root refinements for univariate polynomial equations*`arXiv:1211.4332` (2012)