Algorithm 1010: Boosting Efficiency in Solving Quartic Equations with No Compromise in Accuracy

ALBERTO GIACOMO ORELLANA and CRISTIANO DE MICHELE, Dipartimento di Fisica, "Sapienza" Università di Roma, Italy

Aiming to provide a very accurate, efficient, and robust quartic equation solver for physical applications, we have proposed an algorithm that builds on the previous works of P. Strobach and S. L. Shmakov. It is based on the decomposition of the quartic polynomial into two quadratics, whose coefficients are first accurately estimated by handling carefully numerical errors and afterward refined through the use of the Newton-Raphson method. Our algorithm is very accurate in comparison with other state-of-the-art solvers that can be found in the literature, but (most importantly) it turns out to be very efficient according to our timing tests. A crucial issue for us is the robustness of the algorithm, i.e., its ability to cope with the detrimental effect of round-off errors, no matter what set of quartic coefficients is provided in a practical application. In this respect, we extensively tested our algorithm in comparison to other quartic equation solvers both by considering specific extreme cases and by carrying out a statistical analysis over a very large set of quartics. Our algorithm has also been heavily tested in a physical application, i.e., simulations of hard cylinders, where it proved its absolute reliability as well as its efficiency.

CCS Concepts: • **Mathematics of computing → Solvers**; **Nonlinear equations**; **Computations on polynomials**;

Additional Key Words and Phrases: Quartic equation, factorization into quadratics, Newton-Raphson scheme, numerical solver design, performance

ACM Reference format:
Alberto Giacomo Orellana and Cristiano De Michele. 2020. Algorithm 1010: Boosting Efficiency in Solving Quartic Equations with No Compromise in Accuracy. *ACM Trans. Math. Softw.* 46, 2, Article 20 (May 2020), 28 pages.
https://doi.org/10.1145/3386241

1 INTRODUCTION

The need for solving a quartic equation often arises in the scientific literature. From the Scopus database, by searching for the words "quartic equation," the resulting number of papers since the year 1869 is close to 400. If one looks at the distribution of these papers by subject area, as shown in Figure 1, it is not surprising that the overwhelming majority of the papers are in physics, mathematics, engineering, and computer science, but it is quite surprising to find papers that resort to the solution of a quartic equation in medicine, social sciences, and psychology.

Authors' addresses: A. G. Orellana and C. De Michele, Dipartimento di Fisica, "Sapienza" Università di Roma, P.le A. Moro, 2, Rome, I-00185, Italy; emails: orellana.167635@studenti.uniroma1.it, cristiano.demichele@uniroma1.it.

0098-3500/2020/05-ART20 $15.00
https://doi.org/10.1145/3386241

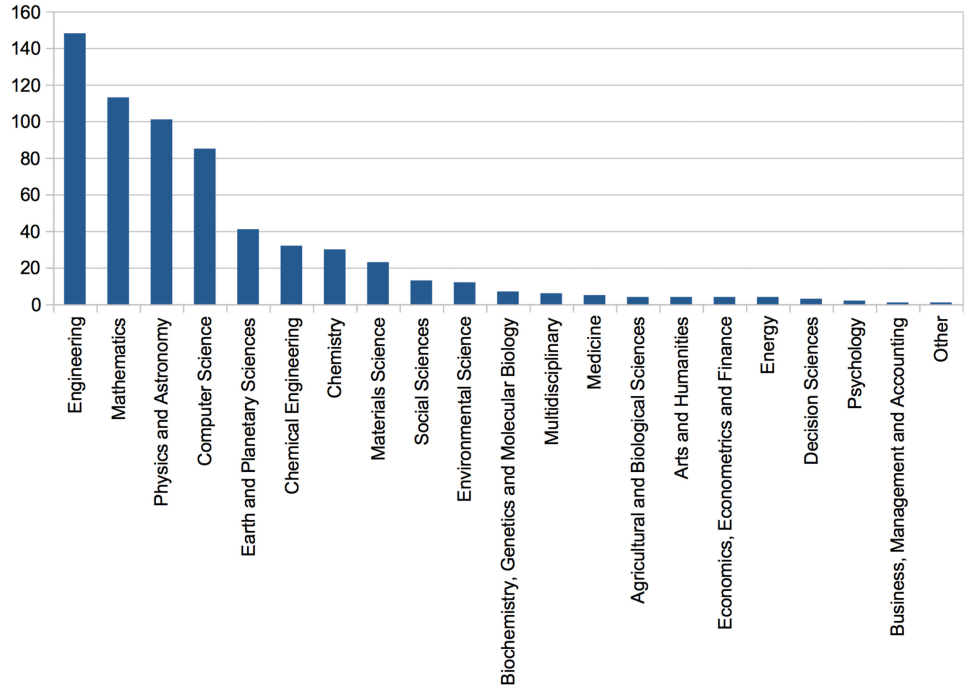

Fig. 1. Distribution by subject area of papers, where "quartic" and "equation" words can be found in the text, from Scopus database (https://www.scopus.com).

In most of these papers, roots of a very large number of quartics have to be accurately calculated. For example, in [60], 2D simulations of hard ellipses are carried out and the overlap test of two ellipses is based on the solution of a quartic equation. Hence, the efficiency of the quartic solver is crucial for simulation timings, but the accuracy of the calculated roots is also very important to prevent unphysical microscopic states of overlapping particles. The need for a very accurate and very efficient (i.e., very fast) quartic solver emerges also in many applications in robotics [40], GPS navigation systems [22, 44, 61], aeronautics [20], steganography [36], acoustics [21], electrical engineering [59], astrophysics [11, 18], particle physics [50], and laser ray tracing [32]. Concerning computer simulations, a very efficient algorithm for simulating hard cylinders that relies on the solution of a quartic equation has been recently proposed by us [43].

The exact analytical solution of quartics dates back to the 16th century; it was discovered by the Italian mathematician Lodovico Ferrari [1, 12, 46]. Although this analytical solution is rather efficient and easy to implement, it cannot be used in practice because it is prone to numerical errors [24]. Many methods have been suggested in the past to overcome the limitations of the original analytic solution [5, 7, 15, 24, 41, 49, 52, 53, 57]. In [15, 41, 49, 57], alternative analytic solutions are proposed that still suffer from the same numerical problems (due to cancellation errors) as the original Ferrari's solution [24]. A different approach in [24], pursuing an iterative method, applies rescaling and deflation of the quartic polynomial, followed by a standard Newton-Raphson approach. The latter method is rather accurate but in terms of efficiency is still much slower than analytic solutions.

Many iterative methods have also been developed that aim to calculate all the roots of a general polynomial [6, 8, 25, 29, 31, 38, 39, 45], but their efficiency is not comparable with that of the

methods specifically designed for quartics. As a paradigmatic example we consider in the present article the method based on the calculation of the eigenvalues of a companion matrix as described in [45]. The latter method turned out to be much slower than the other algorithms we considered. The same conclusions are drawn in [24] for the popular Jenkins-Traub algorithm [30, 31].

Since analytic solutions are numerically inaccurate and accurate methods are much slower than analytic solutions, to carry out simulations of hard cylinders, it was imperative for us to have an algorithm that was both accurate and fast (see the related discussion in the conclusions). Here, building on the idea of a decomposition of the quartic equation into two quadratics, which is inherent in all the classical methods since Cardano (1545) and which is taken up in many papers [2, 4, 5, 7, 9, 13, 14, 23, 26, 51–53], we propose a quartic equation solver that is easy to implement, has an efficiency comparable to that of analytic solutions, and is numerically robust, i.e., very resilient to round-off errors. In our method, the coefficients of the two quadratics into which the quartic is decomposed are first accurately estimated by carefully handling numerical errors, and then they are refined by the Newton-Raphson method [52]. A detailed description of the algorithm will be provided in Section 2.

In Section 3, we test our quartic solver in comparison with the following five state-of-the-art equation solvers, which have been chosen among the ones discussed so far: Flocke's algorithm (FLO) as discussed in [24]; Strobach's algorithm (STR) proposed in [53]; Ferrari's solution (FER) of quartic equations as described, for example, in [1, 33]; another algorithm proposed by Strobach (FQS) in [52]; and finally a quartic equation solver (HQR) discussed in the Numerical Recipe book [45]. The FLO iterative algorithm is based on the use of the Newton-Raphson method and according to the accuracy tests provided in [24] is rather accurate. The iterative methods are typically slower than the analytic solutions, and this suggests that better results in terms of efficiency can be achieved. Our approach is mainly based on the STR algorithm, which is also rather efficient. Anyway, when we first implemented the STR algorithm, we found that it provided roots with very large errors for some specific quartics (see Section 3 for more details). The FER analytic algorithm is based on the venerable analytic solution of Lodovico Ferrari, and as also discussed in [49], it strongly suffers from numerical errors, thus providing very inaccurate roots. This algorithm has to be intended as a paradigmatic example of analytic approaches; in fact, our tests show that Shmakov's particular method [49] (which is the most accurate analytic method proposed so far) is as inaccurate as the FER algorithm. The algorithm FQS proposed in [52] calculates the roots of the quartic by refining the FER analytic solution; it is rather efficient, but we obtained very large errors in some specific cases, as well as for the STR algorithm. Finally, the HQR algorithm is an iterative method based on the calculation of a companion matrix of the quartic equation. This approach is general and not specific for quartics since it can be used to find roots of polynomials with real coefficients of arbitrary degree. The drawback of its generality is that it is rather slow and here it has been employed as a paradigmatic example of iterative methods devoted to the solution of polynomials with real coefficients of arbitrary degree.

We test all these algorithms in some specific extreme cases and we carry out an extensive statistical analysis over a very large set of randomly generated quartic polynomials. This statistical analysis aims to assess the robustness of the quartic solvers, that is, to test whether they are able to keep numerical errors below an acceptable threshold for every arbitrary choice of the quartic coefficients within a realistic domain (i.e., by choosing the quartic coefficients that are expected to occur in realistic practical applications). We also evaluate the efficiency of our algorithm by carrying out timing tests for all the quartic equation solvers considered.

Finally, in Section 4, conclusions will be drawn.

2 METHODS

In this section, we describe in detail our quartic equation solver, which mainly builds on the works of Peter Strobach and Sergei L. Shmakov in [53] and [49], respectively. In [53], an alternative derivation of Shmakov's particular method is provided attempting to minimize numerical errors. Here, we propose a slightly different derivation of Shmakov's approach from the one in [53] with the aim of improving both accuracy and efficiency.

In [53], a quartic equation is solved analytically by exploiting the $\mathbf{LDL}^{\mathrm{t}}$ decomposition. The main issue of this derivation is that the $\mathbf{LDL}^{\mathrm{t}}$ decomposition cannot be straightforwardly used under some conditions (i.e., for any quartic) and here we propose a strategy to overcome this limitation. Since our method is based on an analytic solution, it is not iterative. All the analytic calculations are carefully carried out to alleviate numerical errors, and in addition we exploit the Newton-Raphson method to refine some results. We expect our method to have an efficiency comparable to purely analytic methods and at the same time a much better accuracy.

For the sake of completeness, we include the description of all the numerical techniques that we employed, such as the analytic calculation of the dominant root of a cubic equation and the NR method of solving nonlinear sets of equations.

2.1 A Quartic Equation Solver

Consider a general quartic polynomial in its monic form, i.e.,

$$P(x) = x^4 + ax^3 + bx^2 + cx + d, \tag{1}$$

which can be also written in matrix form as follows:

$$P(x) = \mathbf{y}^{\mathrm{t}}\mathbf{M}_0\mathbf{y}, \tag{2}$$

where $\mathbf{M}_0$ is the following symmetric matrix:

$$\mathbf{M}_0 = \begin{pmatrix} 1 & \frac{1}{2}a & \frac{1}{6}b \\ \frac{1}{2}a & \frac{2}{3}b & \frac{1}{2}c \\ \frac{1}{6}b & \frac{1}{2}c & d \end{pmatrix} \tag{3}$$

and

$$\mathbf{y} = \begin{pmatrix} x^2 \\ x \\ 1 \end{pmatrix}. \tag{4}$$

We note that the following symmetric matrix

$$\mathbf{M}_1 = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 0 & -1 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix} \tag{5}$$

is such that

$$\mathbf{y}^{\mathrm{t}}\mathbf{M}_1\mathbf{y} = 0 \tag{6}$$

so that the matrix

$$\mathbf{M} = \mathbf{M}_0 + \phi\mathbf{M}_1, \tag{7}$$

where $\phi \in \mathbb{R}$, yields the same quartic polynomial as Equation (2), i.e.,

$$P(x) = \mathbf{y}^{\mathrm{t}}\mathbf{M}\mathbf{y}. \tag{8}$$

In the spirit of $\mathbf{LDL}^t$ decomposition [25], if we define the matrices:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 \\ l_1 & 1 & 0 \\ l_3 & l_2 & 1 \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{pmatrix}, \tag{9}$$

the matrix $\mathbf{M}' = \mathbf{LDL}^t$ is symmetric, as is $\mathbf{M}$, and hence we can calculate l_1, l_2, l_3, d_1, d_2, d_3 by requiring that $\mathbf{M} = \mathbf{M}'$, which constitutes the following set of six equations in six unknowns, i.e.:

$$l_1 = \frac{a}{2} \tag{10}$$

$$\frac{c}{2} = d_2 l_2 + l_1 l_3 \tag{11}$$

$$l_3 = \frac{b}{6} + \frac{\phi}{2} \tag{12}$$

$$d_1 = 1 \tag{13}$$

$$\frac{2}{3} b = d_2 + l_1^2 + \phi \tag{14}$$

$$d = d_3 + d_2 l_2^2 + l_3^2, \tag{15}$$

whose explicit solution is

$$l_1 = \frac{a}{2} \tag{16}$$

$$l_2 = -\frac{ab - 6c + 3a\phi}{8b - 3a^2 - 12\phi} \tag{17}$$

$$l_3 = \frac{1}{6}(b + 3\phi) \tag{18}$$

$$d_1 = 1 \tag{19}$$

$$d_2 = \frac{1}{12}(8b - 3a^2 - 12\phi) \tag{20}$$

$$d_3 = 12\,\frac{\det(\mathbf{M})}{8b - 3a^2 - 12\phi}, \tag{21}$$

where

$$\det(\mathbf{M}) = \frac{1}{4}\left[\phi^3 + \left(ac - \frac{b^2}{3} - 4d\right)\phi - \frac{2}{27}b^3 + \frac{1}{3}abc - c^2 - a^2 d + \frac{8}{3}bd\right]. \tag{22}$$

The decomposition of the matrix $\mathbf{M}$ into the $\mathbf{LDL}^t$ form can be achieved as long as a solution of Equations (10) through (15) exists. According to Equations (16) through (21), this solution exists if $d_2 \neq 0$ (the special case $d_2 = 0$ will be discussed below).

To ensure optimal numerical accuracy of the solution provided by Equations (16) through (21), some care must be taken. For calculating l_1 and l_3, we can safely use Equations (16) and (18), but still l_2 and d_2 can be affected by cancellation errors. To make the effect of numerical errors on l_2 and d_2 less severe, some special care is required in their calculation. Since ϕ can be an arbitrary real number, we can choose ϕ as the real root ϕ_0 of largest absolute value (i.e., the dominant root) of the following depressed cubic equation:

$$\det(\mathbf{M}) = 0, \tag{23}$$

which is the same cubic resolvent proposed in [49]. The choice of ϕ as a root of Equation (23) ensures that the quartic equation can be factorized in the product of two quadratic equations, as will be shown in the following. Among the three possible roots, the largest one (i.e., the dominant root) has been chosen since it is expected to be relatively *well conditioned* with respect to the other two.

A method to find the largest root of Equation (23) very accurately is discussed in Section 2.2. As a consequence of Equation (23), if $d_2 \neq 0$, $d_3 = 0$ and Equation (15) becomes

$$d = d_2 l_2^2 + l_3^2. \tag{24}$$

We can thus calculate l_2 and d_2 from Equations (14), (11), and (24), i.e.,

$$d_2 = \frac{2}{3}b - \phi - l_1^2 \tag{25}$$

$$l_2 = \frac{\delta_2}{2d_2} \tag{26}$$

$$\frac{l_2\delta_2}{2} = d - l_3^2, \tag{27}$$

where $\delta_2 = c - al_3$.

Since we have three equations that are satisfied by the two unknowns l_2 and d_2, we have three ways to estimate them, i.e.,

$$\begin{aligned} d_2^{(1)} &= \frac{2}{3}b - \phi - l_1^2 & l_2^{(1)} &= \frac{\delta_2}{2d_2^{(1)}} \\ d_2^{(2)} &= \frac{\delta_2}{2l_2^{(2)}} & l_2^{(2)} &= 2\frac{d - l_3^2}{\delta_2} \\ d_2^{(3)} &= \frac{2}{3}b - \phi - l_1^2 & l_2^{(3)} &= 2\frac{d - l_3^2}{\delta_2} \end{aligned} \tag{28}$$

Assuming that we discard the solutions for which the denominator is 0, the most accurate estimate of l_2 and d_2 will be the one that most accurately satisfies all the Equations (25) through (27); i.e., if we define the function

$$\epsilon_l(l_2, d_2) = \epsilon_0 + \epsilon_1 + \epsilon_2, \tag{29}$$

where

$$\begin{aligned} \epsilon_0 &= \begin{cases} |d_2 + l_1^2 + 2l_3| & \text{if } b = 0 \\ |(d_2 + l_1^2 + 2l_3 - b)/b| & \text{otherwise} \end{cases} \\ \epsilon_1 &= \begin{cases} |2d_2 l_2 + 2l_1 l_3| & \text{if } c = 0 \\ |(2d_2 l_2 + 2l_1 l_3 - c)/c| & \text{otherwise} \end{cases} \\ \epsilon_2 &= \begin{cases} |d_2 l_2^2 + l_3^2| & \text{if } d = 0 \\ |(d_2 l_2^2 + l_3^2 - d)/d| & \text{otherwise,} \end{cases} \end{aligned} \tag{30}$$

we can choose the l_2 and d_2 pair that minimizes ϵ_l. This procedure is intended to mitigate the effect of cancellation errors that can occur in the calculation of both l_2 and d_2.

Having an accurate estimate for l_1, l_2, l_3, and d_2, if we plug the matrix $\mathbf{M}'$ into Equation (8), one has

$$P(x) = (\mathbf{y}^t\mathbf{L})\mathbf{D}(\mathbf{L}^t\mathbf{y}) = d_3 + d_2(l_2 + x)^2 + d_1(l_3 + l_1 x + x^2)^2, \tag{31}$$

but, if $d_2 \neq 0$, from Equation (23) it follows that $d_3 = 0$, and thus we have

$$P(x) = d_2(l_2 + x)^2 + d_1(l_3 + l_1 x + x^2)^2. \tag{32}$$

Hence, the quartic equation

$$P(x) = x^4 + ax^3 + bx^2 + cx + d = 0 \tag{33}$$

now reads

$$(l_3 + l_1 x + x^2)^2 = -d_2(l_2 + x)^2 \tag{34}$$

since $d_1 = 1$.

Taking the square root of both sides of Equation (34), one ends up with two quadratic equations:

$$\begin{aligned} p_1(x) &= x^2 + \alpha_1 x + \beta_1 = 0 \\ p_2(x) &= x^2 + \alpha_2 x + \beta_2 = 0, \end{aligned} \tag{35}$$

where the coefficients α_1, β_1, α_2, and β_2 depend on l_1, l_2, l_3, and d_2 and whose explicit evaluation will be discussed below. Since the four roots, which can be obtained by solving Equation (35), are the four roots of the quartic polynomial, from the fundamental theorem of algebra one has that

$$P(x) = p_1(x)p_2(x). \tag{36}$$

For the calculation of α_1, β_1, α_2, and β_2, we can consider three distinct cases depending on the value of d_2.

Case 1 ($d_2 < 0$). In this case, one has

$$\alpha_1 = l_1 + \sqrt{-d_2} \tag{37}$$

$$\beta_1 = l_3 + \sqrt{-d_2}\, l_2 \tag{38}$$

$$\alpha_2 = l_1 - \sqrt{-d_2} \tag{39}$$

$$\beta_2 = l_3 - \sqrt{-d_2}\, l_2. \tag{40}$$

Since these coefficients can be affected by cancellation errors, we need to find a strategy to improve their estimate. By plugging Equation (35) into Equation (36) and equating the coefficients of the resulting polynomial to those of the polynomial in Equation (1), one has the following set of equations:

$$\beta_1\beta_2 - d = 0 \tag{41}$$

$$\beta_1\alpha_2 + \alpha_1\beta_2 - c = 0 \tag{42}$$

$$\beta_1 + \alpha_1\alpha_2 + \beta_2 - b = 0 \tag{43}$$

$$\alpha_1 + \alpha_2 - a = 0. \tag{44}$$

If $|\beta_2| \leq |\beta_1|$, we use the estimate of β_1 in Equation (38) and from Equation (41) we calculate β_2, i.e.,

$$\beta_2 = \frac{d}{\beta_1}; \tag{45}$$

otherwise, if $|\beta_2| > |\beta_1|$, we use the estimate of β_2 in Equation (40) and we calculate β_1 from Equation (41), i.e.:

$$\beta_1 = \frac{d}{\beta_2}. \tag{46}$$

Concerning α_1 and α_2, if $|\alpha_1| \leq |\alpha_2|$, we assume that the value of α_2 from Equation (39) is accurate, and from Equations (42), (43), and (44) we can calculate three different estimates $\alpha_1^{(i)}$ (with $i = 1, 2, 3$) of α_1, i.e.:

$$\begin{aligned}\alpha_1^{(1)} &= \frac{c - \beta_1\alpha_2}{\beta_2} \\ \alpha_1^{(2)} &= \frac{b - \beta_2 - \beta_1}{\alpha_2} \\ \alpha_1^{(3)} &= a - \alpha_2. \end{aligned} \tag{47}$$

If $\beta_2 = 0$ or $\alpha_2 = 0$, we discard the corresponding estimate for α_1 and we choose the estimate $\alpha_1^{(i)}$ by which we may better reconstruct the coefficients of the quartic polynomial in Equation (1); i.e., given the function

$$\epsilon_q(\alpha_1, \beta_1, \alpha_2, \beta_2) = \epsilon_a + \epsilon_b + \epsilon_c, \tag{48}$$

where

$$\epsilon_a = \begin{cases} |\alpha_1 + \alpha_2| & \text{if } a = 0 \\ |(\alpha_1 + \alpha_2 - a)/a| & \text{otherwise} \end{cases} \tag{49}$$

$$\epsilon_b = \begin{cases} |\beta_1 + \alpha_1\alpha_2 + \beta_2| & \text{if } b = 0 \\ |(\beta_1 + \alpha_1\alpha_2 + \beta_2 - b)/b| & \text{otherwise} \end{cases} \tag{50}$$

$$\epsilon_c = \begin{cases} |\beta_1\alpha_2 + \alpha_1\beta_2| & \text{if } c = 0 \\ |(\beta_1\alpha_2 + \alpha_1\beta_2 - c)/c| & \text{otherwise,} \end{cases} \tag{51}$$

we choose the $\alpha_1^{(i)}$ that minimizes ϵ_q and we set $\alpha_1 = \alpha_1^{(i)}$. If $|\alpha_1| > |\alpha_2|$, we consider α_1 trustworthy, and from Equations (42), (43), and (44) we can calculate the following three estimates $\alpha_2^{(i)}$, with $i = 1, 2, 3$ of α_2:

$$\begin{aligned}\alpha_2^{(1)} &= \frac{c - \alpha_1\beta_2}{\beta_1} \\ \alpha_2^{(2)} &= \frac{b - \beta_2 - \beta_1}{\alpha_1} \\ \alpha_2^{(3)} &= a - \alpha_1. \end{aligned} \tag{52}$$

Again, if $\beta_1 = 0$ or $\alpha_1 = 0$, we discard the corresponding estimates, and we select the one that minimizes the function ϵ_q.

Case 2 ($d_2 > 0$). In this case, the p_1 and p_2 coefficients are

$$\alpha_1 = l_1 + i\sqrt{d_2} \tag{53}$$

$$\beta_1 = l_3 + i\sqrt{d_2}\, l_2 \tag{54}$$

$$\alpha_2 = l_1 - i\sqrt{d_2} \tag{55}$$

$$\beta_2 = l_3 - i\sqrt{d_2}\, l_2. \tag{56}$$

Note that there are no cancellations errors in carrying out the sums for calculating α_1, β_1, α_2, and β_2 from Equations (53) through (56) since in all cases one addend is a real number and the other is a purely imaginary complex number; hence, we can safely use these estimates without any special care.

Case 3 ($d_2 = 0$). From Equations (20) and (21) we have

$$d_3(\phi_0) = \frac{\det(\mathbf{M})}{d_2(\phi_0)}; \tag{57}$$

therefore, if $d_2(\phi_0) = 0$, d_3 is not defined and we cannot resort to Equation (34) for calculating the quartic roots. A different strategy must be devised.

We start by noting that (i) according to Equation (11), $\lim_{\phi\to\phi_0} d_2(\phi)l_2(\phi)$ is not infinite, and that (ii) according to Equation (57), $\lim_{\phi\to\phi_0} d_3(\phi)$ is also not infinite (because ϕ_0 is a root of both $d_2(\phi)$ and $\det[\mathbf{M}(\phi)]$ polynomials). In view of (i) and (ii), Equation (15) implies that $\lim_{\phi\to\phi_0} l_2(\phi)$ is not infinite, so that we conclude that $\lim_{\phi\to\phi_0} d_2(\phi)l_2(\phi) = 0$. Then from Equation (31), one has

$$P(x) = \lim_{\phi\to\phi_0} (\mathbf{y}^t\mathbf{L})\mathbf{D}(\mathbf{L}^t\mathbf{y}) = d_3 + (l_3 + l_1x + x^2)^2, \tag{58}$$

where from Equation (15) $d_3 = d - l_3^2$ and l_1 and l_3 have the same values as those we have already calculated using Equations (16) and (18). The four roots of the quartic equation can be found by solving the equation

$$d_3 + (l_3 + l_1x + x^2)^2 = 0. \tag{59}$$

By taking the square root of both sides of Equation (59), one ends up with the two quadratic equations, which are identical to Equation (35) except that in this case the coefficients of $p_1(x)$ and $p_2(x)$ are

$$\begin{aligned} \alpha_1 &= l_1 \\ \beta_1 &= l_3 + \sqrt{-d_3} \\ \alpha_2 &= l_1 \\ \beta_2 &= l_3 - \sqrt{-d_3}, \end{aligned} \tag{60}$$

where all of them have to be real, i.e., $d_3 \le 0$; otherwise, one would obtain complex root pairs that are not conjugate. The same conclusion about the sign of d_3 can be drawn by the following reasoning. Since ϕ_0 is a root of both $\det(\mathbf{M})$ and $d_2(\phi)$, from Equation (57) one has that

$$d_3(\phi_0) = \lim_{\phi\to\phi_0} \frac{\det(\mathbf{M})}{d_2(\phi)} = -\frac{1}{4}(\phi_0 - \phi_1)(\phi_0 - \phi_2), \tag{61}$$

where ϕ_1 and ϕ_2 are the two other roots of Equation (23). We can consider two possible cases depending on whether the roots ϕ_1 and ϕ_2 are (i) real or (ii) complex conjugate. In the case (i), since we chose ϕ_0 to be the dominant real root (i.e., the real root with the largest absolute value) of Equation (23), $\phi_0 - \phi_1$ and $\phi_0 - \phi_2$ are both positive or both negative and one necessarily has that $d_3 \le 0$. In the case (ii), $\phi_2 = \phi_1^*$ and

$$d_3(\phi_0) = -\frac{1}{4}\left(\phi_0^2 - 2\,\mathrm{Re}(\phi_1)\phi_0 + |\phi_1|^2\right), \tag{62}$$

which is a quadratic function in ϕ_0 with downward concavity. The maximum value of this quadratic function is $\frac{1}{4}\{[\mathrm{Re}(\phi_1)]^2 - |\phi_1|^2\}$, which is negative $\forall\phi_1$, and hence we conclude again that $d_3(\phi_0) \le 0$.

The coefficients β_1 and β_2 can be affected by cancellation errors. To improve their estimate, if $|\beta_1| > |\beta_2|$, then we trust β_1 and set

$$\beta_2 = \frac{d}{\beta_1}, \tag{63}$$

while if $|\beta_2| > |\beta_1|$, β_2 is considered reliable and set

$$\beta_1 = \frac{d}{\beta_2}. \tag{64}$$

We note that if $d_2 \neq 0$ but $d_2 \approx 0$, the estimates of l_2 and d_2 through Equations (25) and (26) can be significantly affected by cancellation errors. According to Equation (25), where d_2 is expressed as a sum of $2b/3$, $-\phi$, and $-l_1^2$, and to the discussion of termination criteria for finding zeros of polynomials in [28], we can consider $d_2 \approx 0$ if the following condition holds:

$$d_2 \leq \epsilon_m \max\{|2b/3|, |\phi_0|, l_1^2\}, \tag{65}$$

where $\epsilon_m \approx 2.22045 \times 10^{-16}$ in double precision. Hence, if the condition in Equation (65) is fulfilled, we suggest proceeding according to the following steps:

(1) Let's define
$$\mathcal{S}_1 = \{\alpha_1^{(1)}, \beta_1^{(1)}, \alpha_2^{(1)}, \beta_2^{(1)}\} \tag{66}$$
as the set of coefficients that have been obtained in either *Case 1* or *Case 2* (depending on the sign of d_2).

(2) Calculate the set of coefficients
$$\mathcal{S}_2 = \{\alpha_1^{(2)}, \beta_1^{(2)}, \alpha_2^{(2)}, \beta_2^{(2)}\} \tag{67}$$
according to *Case 3*.

(3) Given the following function, which quantifies the accuracy of a set of coefficients α_1, β_1, α_2, and β_2 (where the most accurate set is the one that better reconstructs the coefficients a, b, c, and d of the quartic polynomial),
$$\tilde{\epsilon}_q(\alpha_1, \beta_1, \alpha_2, \beta_2) = \epsilon_a + \epsilon_b + \epsilon_c + \epsilon_d, \tag{68}$$
where ϵ_a, ϵ_b, ϵ_c are as in Equations (49) through (51) and
$$\epsilon_d = \begin{cases} |\beta_1\beta_2| & \text{if } d = 0 \\ |(\beta_1\beta_2 - d)/d| & \text{otherwise;} \end{cases} \tag{69}$$
we find its value for the set $\mathcal{S}_1$ and $\mathcal{S}_2$ and choose the set that provides the smallest $\tilde{\epsilon}_q$.

Note that if $d = 0$, the quartic has one root that is exactly 0 and the other three roots can be better obtained by solving the following cubic equation: $x^3 + ax^2 + bx + c = 0$. This check could be done at the very beginning of the computation.

Before calculating the roots of $p_1(x)$ and $p_2(x)$ (i.e., the roots of the quartic equation), it is advisable to refine the coefficients α_1, β_1, α_2, and β_2 by using the Newton-Raphson (NR) method to solve Equations (41) through (44). The NR method will be extensively discussed in Section 2.3. Recalling that the roots of the quartic equation are obtained by finding the roots of $p_1(x)$ and $p_2(x)$ (see Equations (35) and (36)), if the coefficients α_i and β_i are real (as in *Case 1* or *Case 3* above), the four roots x_1, x_2, x_3, and x_4 of the quartic equation can be calculated as follows [45]:

(1) Set $k = 1$.
(2) Calculate $\Delta = \alpha_k^2 - 4\beta_k$.
(3) Set $t = 2(k-1)$.
(4) If $\Delta < 0$, calculate the two roots x_{t+1} and x_{t+2} as
$$\begin{aligned} x_{t+1} &= -\frac{\alpha_k}{2} + \frac{1}{2}\sqrt{-\Delta}\, i \\ x_{t+2} &= -\frac{\alpha_k}{2} - \frac{1}{2}\sqrt{-\Delta}\, i, \end{aligned} \tag{70}$$
and if $k = 2$, terminate; otherwise, set $k = 2$ and go to step (2).

(5) If $\Delta \geq 0$, calculate

$$\eta_M = \begin{cases} -\frac{\alpha_k}{2} - \frac{\sqrt{\Delta}}{2} & \text{if } \alpha_k \geq 0 \\ -\frac{\alpha_k}{2} + \frac{\sqrt{\Delta}}{2} & \text{otherwise.} \end{cases} \tag{71}$$

(6) Calculate

$$\eta_m = \begin{cases} 0 & \text{if } \eta_M = 0 \\ \frac{\beta_k}{\eta_M} & \text{otherwise.} \end{cases} \tag{72}$$

(7) The two real roots x_{t+1} and x_{t+2} are

$$\begin{aligned} x_{t+1} &= \eta_M \\ x_{t+2} &= \eta_m. \end{aligned} \tag{73}$$

(8) If $k = 2$, terminate; otherwise, set $k = 2$ and go to step (2).

If the coefficients α_i and β_i with $i = 1, 2$ are complex (as in *Case 2*), the two roots $x_1^{(i)}$ and $x_2^{(i)}$ of each quadratic polynomial $p_i(x)$ can be calculated as described in the following [53]. If we define

$$\eta = \begin{cases} \gamma_1 & \text{if } |\gamma_1| > |\gamma_2| \\ \gamma_2 & \text{otherwise,} \end{cases} \tag{74}$$

where

$$\gamma_1 = -\frac{\alpha_i}{2} + \sqrt{\frac{\alpha_i^2}{4} - \beta_i} \tag{75}$$

$$\gamma_2 = -\frac{\alpha_i}{2} - \sqrt{\frac{\alpha_i^2}{4} - \beta_i}, \tag{76}$$

then the two roots of $p_i(x)$ are

$$\begin{aligned} x_1^{(i)} &= \eta \\ x_2^{(i)} &= \frac{\beta_i}{\eta}. \end{aligned} \tag{77}$$

We note that, since the quartic polynomial has real coefficients, the roots $x_1^{(2)}$ and $x_2^{(2)}$ must be the complex conjugates of $x_1^{(1)}$ and $x_2^{(1)}$, i.e., $x_1^{(2)} = (x_1^{(1)})^*$ and $x_2^{(2)} = (x_2^{(1)})^*$. This means that we can carry out the calculations in Equations (75), (76), and (77) just for $i = 1$. The four roots x_1, x_2, x_3, and x_4 of the quartic equation are then

$$\begin{aligned} x_1 &= x_1^{(1)} \\ x_2 &= x_1^* \\ x_3 &= x_2^{(1)} \\ x_4 &= x_3^*. \end{aligned} \tag{78}$$

An alternative method to solve a quadratic equation with complex coefficients is suggested in [45]. We also implemented it, but we did not find any significant difference in the accuracy from our tests discussed in Section 3.

2.2 Dominant Root of the Depressed Cubic

In Section 2.1, ϕ_0 is the dominant root of the following depressed cubic equation:

$$\det[\mathbf{M}(\phi)] = \phi^3 + g\phi + h = 0, \tag{79}$$

where

$$g = ac - 4d - \frac{b^2}{3} \tag{80}$$

$$h = \left(ac + 8d - \frac{2}{9}b^2\right)\frac{b}{3} - c^2 - a^2 d. \tag{81}$$

Minimization of numerical errors in the calculation of ϕ_0 can be achieved by a *shift* of x [49, 53], i.e., by performing the transformation $x' = x - s$ (with $s \in \mathbb{R}$), so that Equation (33) becomes

$$P(x') = x'^4 + a'x^3 + b'x^2 + c'x' + d', \tag{82}$$

where the coefficients in Horner form of this shifted quartic are

$$\begin{aligned} a' &= a + 4s \\ b' &= b + 3s(a + 2s) \\ c' &= c + s[2b + s(3a + 4s)] \\ d' &= d + s\{c + s[b + s(a + s)]\}. \end{aligned} \tag{83}$$

It can be seen that the associated depressed cubic equation, i.e.,

$$\phi^3 + g'\phi + h' = 0, \tag{84}$$

with

$$g' = a'c' - 4d' - \frac{b'^2}{3} \tag{85}$$

$$h' = \left(a'c' + 8d' - \frac{2}{9}b'^2\right)\frac{b'}{3} - c'^2 - a'^2 d' \tag{86}$$

is the same as the one in Equation (79), i.e., $g = g'$ and $h = h'$. This can easily be proved by plugging into Equations (85) and (86) the coefficients a', b', c', and d', which are defined in Equation (83). Note that, even if $g = g'$ and $h = h'$, the calculation of $g'(s)$ and $h'(s)$ through Equations (85) and (86) provides an effective way to mitigate the detrimental effect of cancellation errors in the calculation of the dominant root of Equation (84) if a judicious choice of s is made. In [53], it is suggested that the best strategy is to choose s so that $b'(s)$ is either exactly 0 or an absolute minimum, i.e., to choose s in such a way that

$$\begin{cases} b'(s) = b + 3as + 6s^2 = 0 & \text{if } 9a^2 - 24b \geq 0 \\ \frac{db'}{ds} = 3a + 12s = 0 & \text{otherwise,} \end{cases} \tag{87}$$

where we use the root of smallest magnitude in the quadratic equation, so that

$$s = \begin{cases} \frac{-2b}{3a + \mathrm{sgn}(a)\sqrt{9a^2 - 24b}} & \text{if } 9a^2 - 24b \geq 0 \\ -\frac{a}{4} & \text{otherwise.} \end{cases} \tag{88}$$

Having set the coefficients g' and h' of Equation (84), we are left with the task of finding its dominant root. An analytic and accurate solution of a generic cubic equation is provided in Section 5.6

of [45]. Here we only need the dominant real root (i.e., the real root with largest absolute value); hence, given

$$Q = -\frac{g'}{3}$$
$$R = \frac{h'}{2}, \tag{89}$$

the dominant root ϕ_0 of Equation (84) can be obtained according to the following two possible cases:

Case 1: $R^2 < Q^3$. If we define

$$\theta = \arccos\left(\frac{R}{\sqrt{Q^3}}\right), \tag{90}$$

where we assume that the function $\arccos(x)$ returns values in the range $[0, \pi]$, the dominant root is

$$\phi_0 = \begin{cases} -2\sqrt{Q}\cos\left(\frac{\theta}{3}\right) & \text{if } \theta < \frac{\pi}{2} \\ -2\sqrt{Q}\cos\left(\frac{\theta+2\pi}{3}\right) & \text{otherwise.} \end{cases} \tag{91}$$

Case 2: $R^2 \geq Q^3$. In this case the dominant root ϕ_0 is

$$\phi_0 = A + B, \tag{92}$$

where

$$A = -\operatorname{sgn}(R)\left(|R| + \sqrt{R^2 - Q^3}\right)^{\frac{1}{3}}$$

and

$$B = \begin{cases} 0 & \text{if } A = 0 \\ Q/A & \text{otherwise.} \end{cases} \tag{93}$$

The above analytical estimate of the dominant root cannot handle large absolute values of Q or R (i.e., in double precision $|Q| > 10^{102}$ or $|R| > 10^{154}$); hence, in this case, one can recast Equations (91) and (92) and express them in terms of the ratio Q/R to reduce the risk of overflows.

Given

$$K = \begin{cases} 1 - Q\left(\frac{Q}{R}\right)^2 & \text{if } |Q| < |R| \\ \operatorname{sgn}(Q)\left[\left(\frac{R}{Q}\right)^2 \frac{1}{Q} - 1\right] & \text{otherwise,} \end{cases} \tag{94}$$

the roots of the depressed cubic can be obtained according to the following three possible cases if $|Q|$ or $|R|$ are large.

Case 1: $R = 0$. The dominant root ϕ_0 is simply

$$\phi_0 = \begin{cases} 0 & \text{if } g' > 0 \\ \sqrt{-g'} & \text{otherwise.} \end{cases} \tag{95}$$

Case 2: $K < 0$. The dominant root ϕ_0 can be obtained according to Equation (91), where θ is replaced by

$$\theta = \arccos\left(\frac{R}{Q}\frac{1}{\sqrt{Q}}\right).$$

Case 3: $K \geq 0$. Use Equations (92) to calculate ϕ_0 but replace A with

$$A = \begin{cases} -\operatorname{sgn}(R)\left[|R|\left(1 + \sqrt{K}\right)\right]^{\frac{1}{3}} & \text{if } |Q| < |R| \\ -\operatorname{sgn}(R)\left(|R| + \sqrt{|Q|}|Q|\sqrt{K}\right)^{\frac{1}{3}} & \text{otherwise.} \end{cases} \tag{96}$$

We can further improve the estimate of ϕ_0 by employing the Newton-Raphson method, i.e.,

(1) Set $x = \phi_0$.
(2) Calculate $f = (x^2 + g')x + h'$.
(3) If $|f| < \epsilon_m \max\{|x^3|, |g'x|, |h'|\}$, then terminate.
(4) Calculate $\delta f = 3x^2 + g'$.
(5) If $\delta f = 0$, terminate.
(6) Set $x_o = x$ and $f_o = f$ and
(7) Update x as follows: $x \leftarrow x - \frac{f}{\delta f}$.
(8) Calculate the new value of f; i.e., set $f = (x^2 + g')x + h'$.
(9) If $f = 0$, terminate.
(10) If $|f| > |f_o|$, set $x = x_o$ and terminate.
(11) Go to step (4).

If with the above procedure one obtains a dominant root of the depressed cubic that is infinite or an undefined/unrepresentable number (NAN, which stands for *not a number*), one can employ a simple polynomial rescaling. Given a very large number k_q (in our code we use $k_q = 7.16 \times 10^{76}$, which is suitable for double floating-point numbers), rescaled polynomial coefficients a_s, b_s, c_s, and d_s are calculated as follows:

$$\begin{aligned} a_s &= a/k_q \\ b_s &= b/k_q^2 \\ c_s &= c/k_q^3 \\ d_s &= d/k_q^4. \end{aligned} \tag{97}$$

These coefficients are then used to calculate the roots of the corresponding rescaled polynomial, i.e.,

$$P_s(x) = x^4 + a_s x^3 + b_s x^2 + c_s + d_s, \tag{98}$$

as described in Section 2.1, and the roots of the original polynomial in Equation (1) can finally be obtained by multiplying each root of the rescaled polynomial by the factor k_q.

If by employing the above polynomial rescaling one still obtains a dominant root of the depressed cubic that is infinite or NAN, a polynomial rescaling of the depressed cubic can be attempted too. We consider a very large number k_c (in our code we use $k_c = 3.49 \times 10^{102}$, which is suitable for double floating-point numbers) and the coefficients g' and h' in Equation (84) are replaced with g'_s and h'_s, which are calculated as follows:

$$g'_s = a'_s c'_s - \frac{4d'_s}{k_c} - \frac{{b'_s}^2}{3} \tag{99}$$

$$h'_s = \left(a'_s c'_s + \frac{8d'_s}{k_c} - \frac{2{b'_s}^2}{9} \right) \frac{b'_s}{3} - {c'_s}\frac{c'_s}{k_c} - {a'_s}^2 d'_s, \tag{100}$$

where

$$\begin{aligned} a'_s &= \frac{a'}{k_c} \\ b'_s &= \frac{b'}{k_c} \\ c'_s &= \frac{c'}{k_c} \\ d'_s &= \frac{d'}{k_c}. \end{aligned} \tag{101}$$

These coefficients are then used to calculate the dominant root of the following depressed cubic:

$$\phi^3 + g'_s\phi + h'_s = \phi^3 + \frac{g'}{k_c^2}\phi + \frac{h'}{k_c^3} = 0. \tag{102}$$

The dominant root of the original cubic polynomial is then recovered by multiplying the dominant root of the latter equation by k_c.

2.3 Newton-Raphson Method for Refining the Coefficients of p_1 and p_2

As already anticipated, the coefficients α_1, β_1, α_2, and β_2 can be improved by employing a Newton-Raphson method to solve Equations (41) through (44) [52]. Given an initial guess for

$$\mathbf{z} = \begin{pmatrix} \alpha_1 \\ \beta_1 \\ \alpha_2 \\ \beta_2 \end{pmatrix} \tag{103}$$

such as the one obtained in Section 2.1, the NR algorithm, which we employed, is the following:

(1) Calculate the vector $\mathbf{F}$:

$$\mathbf{F} = \begin{pmatrix} \beta_1\beta_2 - d \\ \beta_1\alpha_2 + \alpha_1\beta_2 - c \\ \beta_1 + \alpha_1\alpha_2 + \beta_2 - b \\ \alpha_1 + \alpha_2 - a \end{pmatrix}. \tag{104}$$

(2) Calculate the quantity e_t as follows:

$$e_t = \epsilon_a + \epsilon_b + \epsilon_c + \epsilon_d, \tag{105}$$

where ϵ_a, ϵ_b, ϵ_c, and ϵ_c are as in Equations (49) through (51) and (69).

(3) If $e_t = 0$, terminate.

(4) If $\mathbf{J} = \frac{\partial \mathbf{F}(\alpha_1,\beta_1,\alpha_2,\beta_2)}{\partial(\alpha_1,\beta_1,\alpha_2,\beta_2)}$, i.e., $\mathbf{J}$ is the Jacobian matrix of $\mathbf{F}$, calculate its inverse $\mathbf{J}^{-1}$, which can be written as follows:

$$\mathbf{J}^{-1} = \frac{1}{\det(\mathbf{J})} \begin{pmatrix} C_1 & C_2 & C_3 & -\beta_1 C_2 - \alpha_1 C_3 \\ \alpha_1 C_1 + C_2 & -\beta_1 C_1 & -\beta_1 C_2 & -\beta_1 C_3 \\ -C_1 & -C_2 & -C_3 & \alpha_2 C_3 + \beta_2 C_2 \\ -\alpha_2 C_1 - C_2 & \beta_2 C_1 & \beta_2 C_2 & \beta_2 C_3 \end{pmatrix}, \tag{106}$$

where $C_1 = \alpha_1 - \alpha_2$, $C_2 = \beta_2 - \beta_1$, $C_3 = \beta_1\alpha_2 - \alpha_1\beta_2$, and

$$\det(\mathbf{J}) = \beta_1^2 - \beta_1[\alpha_2(\alpha_1 - \alpha_2) + 2\beta_2] + \beta_2[\alpha_1(\alpha_1 - \alpha_2) + \beta_2].$$

(5) If $\det(\mathbf{J}) = 0$, terminate.

(6) Store the actual value of $\mathbf{z}$, i.e.,

$$\mathbf{z}_o = \mathbf{z}. \tag{107}$$

(7) Update $\mathbf{z}$ as follows:

$$\mathbf{z} \leftarrow \mathbf{z} - \mathbf{J}^{-1}\mathbf{F}. \tag{108}$$

(8) Calculate the new value of $\mathbf{F}$ and e_t according to Equations (104) and (105), respectively.

(9) If $e_t = 0$, terminate

(10) If the new value of e_t is greater than the old one, set $\mathbf{z} = \mathbf{z}_o$ and terminate.

(11) Go to step (2).

We also limit the maximum number of possible iterations to eight, although we observe that no more than two to three iterations are needed at most.

2.4 Quartic Equations with Complex Coefficients

Our algorithm can be straightforwardly generalized to solve quartic polynomials with complex coefficients. The few needed changes are the following ones:

(i) For the estimate of ϕ_0 in Section 2.2, one can always use the smallest-magnitude complex root of the quadratic equation $b'(s) = 0$ (see Equation (87)) and the dominant root of the complex depressed cubic can be obtained as discussed in [45]
(ii) For the calculation of the coefficients of $p_1(x)$ and $p_2(x)$ in Section 2.1, we always proceed according to either *Case 3* if $d_2 = 0$ or *Case 1* if $d_2 \neq 0$ and in the latter case $|d_2|$ must be replaced by d_2 (where we are not taking the absolute value as before) in Equation (37) through (40).
(iii) Since the coefficients of $p_1(x)$ and $p_2(x)$ are complex, their roots can be calculated according to Equation (77). Before calculating the roots of $p_1(x)$ and $p_2(x)$, we suggest using the Newton-Raphson method, as described in Section 2.3, to refine the coefficients α_1, β_1, α_2, and β_2.

The code in C language for solving a quartic equation with complex coefficients is also provided with the manuscript.

3 RESULTS AND DISCUSSION

In this section, we provide a thorough test of our quartic equation solver in comparison with several existing approaches that are available in the literature. We carried out accuracy tests for specific sets of quartics in Section 3.1, statistical tests on a very large set of quartics in Section 3.2, and timing tests in Section 3.3 to assess the efficiency of our quartic equation solver. In the accuracy tests, we consider a set of 24 quartics as shown in Table 1. In each case from 1 to 22, we choose a specific set of roots and we build the quartic coefficients accordingly, while in cases 23 and 24 we provide the quartic coefficients directly. In the accuracy tests from 1 to 22, the roots display various extreme situations, such as widely spread, clustered (i.e., very closely spaced), or even multiple (triple or quadruple) roots. In case 23, the coefficients have been chosen to yield $d_2 = 0$ in our quartic solver, while case 24 is designed to have widely spread quartic coefficients. Statistical tests were performed by generating a very large set of quartic equations to mimic a typical practical situation, such as the calculation of intersection points between a disk and a cylindrical rim [43]. The results obtained from our solver were compared with the ones from the following five quartic equation solvers, already briefly discussed in the Introduction:

FLO This algorithm is described in detail in [24]. The author kindly provided an updated Fortran version of the source code, which we used for our tests. We translated this version to the C language and we checked that our implementation provides the same results as the Fortran version in all tests that we considered in the present study.
STR The source code of this algorithm, which is discussed in [53], is available in Fortran. We translated this code to C language and we checked that our implementation provides the same results as the Fortran version (as we did for FLO algorithm).
FER We also considered the analytic solution of Equation (33) as reported, for example, in [1, 33]. In [49], it is observed that none of the analytic solutions, such as the ones discussed in [1, 15, 33, 41, 49, 57], *would completely suit the needs of stable computations* and a novel analytic solution is proposed to overcome this limitation. We tested the latter

Table 1. Set of 24 Quartics Used for the Accuracy Tests

Case	Roots	Note
1	10^9, 10^6, 10^3, 1	Four widely spaced real
2	2.003, 2.002, 2.001, 2	Four closely spaced small real
3	10^{53}, 10^{50}, 10^{49}, 10^{47}	Four large real
4	10^{14}, 2, 1, -1	One large, three small real
5	-2×10^7, 10^7, 1, -1	Two large, two small real
6	10^7, -10^6, $1\pm i$	Two large real, two small complex
7	-7, -4, $-10^6\pm10^5 i$	Two small real, two large complex
8	10^8, 11, $10^3\pm i$	One large and small real, two medium complex
9	$10^7\pm10^6 i$, $1\pm2i$	Two large, two small complex
10	$10^4\pm3i$, $-7\pm10^3 i$	Four complex, mixed size real and imag. parts
11	$1.001\pm4.998i$, $1\pm5.001i$	Four closely spaced complex
12	$10^3\pm3i$, $10^3\pm i$	Four complex, equal real, small imag. parts
13	$2\pm10^4 i$, $1\pm10^3 i$	Four complex, small real, large imag. parts
14	1000, 1000, 1000, 1000	Quadruple root
15	1000, 1000, 1000, 10^{-15}	Triple and one small root
16	$1\pm0.1i$, $10^{16}\pm10^7 i$	Four complex widely spaced
17	10000, 10001, 10010, 10100	Four closely spaced large real
18	$40000\pm300i$, $30000\pm7000i$	Four quite large complex
19	10^{44}, 10^{30}, 10^{30}, 1	Four very widely spaced real
20	10^{14}, 10^7, 10^7, 1	Four widely spaced real
21	10^{15}, 10^7, 10^7, 1	Four widely spaced real
22	10^{154}, 10^{152}, 10, 1	Two very large real roots
Case	Quartic Coefficients	Note
23	$a=1$, $b=1$, $c=\frac{3}{8}$, $d=10^{-3}$	Four distinct roots with $d_2\approx0$
24	$a=-\left(1+\frac{1}{S}\right)$, $b=\frac{1}{S}-S^2$	
	$c=S^2+S$, $d=-S$ with $S=10^{30}$	Widely spaced coefficients

For the first 12 tests we set the roots of the quartics and we calculate the coefficients accordingly. For tests 23 and 24 we specified the coefficients of the quartics.

novel method carefully and, even if it is as efficient as the FER approach, it turned out to be much less accurate in our accuracy and statistical tests reported in Sections 3.1 and 3.2, respectively. For this reason, we decided to show only the results obtained by the FER analytic solution.

FQS We also coded this algorithm in C for comparison according to the prescriptions that can be found in [52].

HQR This algorithm, described in [45], is not specifically designed for quartic equations, since it can be used to find roots of polynomials of arbitrary degree. Another general purpose algorithm, i.e., the popular Jenkins-Traub algorithm [31], has already been tested in [24] and it is not expected to perform better than HQR in terms of accuracy and efficiency.

We used gcc to compile our C codes with the -O3 optimization flag and we used double-precision floating-point arithmetic (machine accuracy is $\approx 2.22\times10^{-16}$). Timing and accuracy tests were performed on a Macbook Pro 13" laptop with an Intel 3.3GHz dual-core i7 processor, while for statistical tests we used a Linux server with four Intel Xeon E5–4620 v2 2.60GHz octa-core processors. For timing tests on the Macbook Pro 13" laptop we used both the *Apple LLVM version 9.0.0*

Table 2. Maximum Relative Error of All the Algorithms Considered for the Cases from 1 to 22 in Table 1

Case	Maximum Relative Error ε_{max} of Calculated Roots					
	OQS	FLO	STR	FER	FQS	HQR
1	0	1.2×10^{-16}	0	1.6×10^{-8}	0	5.2×10^{-15}
2	8.8×10^{-7}	3.6×10^{-6}	8.8×10^{-7}	2.9×10^{-3}	2.6×10^{-6}	2.2×10^{-6}
3	1.3×10^{-16}	3.9×10^{-16}	NAN	NAN	NAN	8.3×10^{-16}
4	0	0	3.2×10^{-3}	0.5	0	6.1×10^{-11}
5	0	1.9×10^{-16}	0	7.8×10^{-3}	0	4×10^{-15}
6	0	1.2×10^{-16}	0	0	0	1.1×10^{-15}
7	0	0	2.4×10^{-15}	1.6×10^{-7}	0	2.2×10^{-15}
8	0	0	1.1×10^{-13}	0	0	2.9×10^{-12}
9	0	0	1.6×10^{-15}	6.4×10^{-3}	0	6×10^{-15}
10	0	1.5×10^{-17}	0	1.5×10^{-13}	4.4×10^{-18}	2.4×10^{-13}
11	9.4×10^{-14}	2.9×10^{-13}	1.6×10^{-13}	1.4×10^{-13}	8.9×10^{-14}	3×10^{-13}
12	0	0	0	1.9×10^{-3}	1.4×10^{-3}	4.6×10^{-8}
13	0	0	0	0	0	5.5×10^{-16}
14	0	0	1.5×10^{-8}	0	0	1.6×10^{-4}
15	2×10^{-16}	0	2×10^{-16}	1	9×10^{-12}	7×10^{-6}
16	10^{-9}	1.5×10^{-8}	3.5×10^{-8}	3×10^{7}	10^{-9}	1.6×10^{-8}
17	2.5×10^{-7}	3.9×10^{-7}	1.4×10^{-6}	1.6×10^{-3}	7×10^{-8}	1.3×10^{-6}
18	2.7×10^{-16}	2.7×10^{-16}	8.6×10^{-16}	8.9×10^{-12}	0	1.8×10^{-13}
19	1.4×10^{-16}	1.4×10^{-16}	NAN	NAN	NAN	1
20	1.3×10^{-8}	2.2×10^{-16}	1.8×10^{-8}	5.2×10^{-2}	1.3×10^{-8}	7.4×10^{-8}
21	1.1×10^{-16}	1.3×10^{-8}	2.2×10^{-8}	4.2×10^{-1}	1.1×10^{-16}	6.6×10^{-8}
22	1.1×10^{-16}	2×10^{-15}	NAN	NAN	NAN	1

(clang-900.0.39.2) of gcc (apple-gcc) and the gnu gcc version 7.2.0 (gnu-gcc), while on the Linux server we used gcc version 4.8.4. The random numbers in Sections 3.2 and 3.3 were generated by using the C function *drand48*() with the use of the same seed for testing all algorithms.

3.1 Accuracy Test

In Table 1, we show all the cases considered with a brief descriptive note. In the cases from 1 to 22, we use Vieta's formulas to calculate—using quadruple precision to minimize the effect of round-off errors—the quartic polynomial coefficients in Equation (33), i.e.:

$$\begin{aligned} a &= -(x_1 + x_2 + x_3 + x_4) \\ b &= x_1(x_2 + x_3) + x_2(x_3 + x_4) + x_4(x_1 + x_3) \\ c &= -x_1x_2(x_3 + x_4) - x_3x_4(x_1 + x_2) \\ d &= x_1x_2x_3x_4, \end{aligned} \tag{109}$$

where x_1, x_2, x_3, an x_4 are the four roots of the quartic polynomial.

For each set of roots $\{x_i^c\}$ calculated by a quartic equation solver, we calculated the maximum relative error, i.e.:

$$\varepsilon_{max} = \max_i \left\{ \frac{|x_i^c - x_i|}{|x_i|} \right\}, \tag{110}$$

which is shown in Table 2 for all quartic solvers we considered. Our quartic solver (OQS) performs very well, providing exact solutions in 11 cases and keeping the relative error at a minimum in the

Table 3. Roots Obtained from All the Algorithms Considered in Cases 23 and 24 Shown in Table 1

Case	Calculated Roots		
	OQS	FLO	STR
23	$-0.25 + 0.82835034123894i$	$-0.25 + 0.828350341238939i$	$-0.25 + 0.559016994374947i$
	$-0.25 - 0.82835034123894i$	$-0.25 - 0.828350341238939i$	$-0.25 - 0.559016994374947i$
	-0.497314148060048	-0.497314148060049	$-0.25 + 0.559016994374947i$
	-0.00268585193995149	-0.00268585193995149	$-0.25 - 0.559016994374947i$
24	-10^{30}	-10^{30}	$-7.80368846212467 \times 10^{57}$
	10^{30}	10^{30}	1.66666666666667
	1	1	1
	10^{-30}	10^{-30}	10^{-30}
	FER	FQS	HQR
23	$-0.25 + 0.828350341238939i$	$-0.25 - 0.82835034123894i$	$-0.25 + 0.828350341238939i$
	$-0.25 - 0.828350341238939i$	$-0.25 + 0.82835034123894i$	$-0.25 - 0.828350341238939i$
	-0.497314148060048	-0.497314148060048	-0.497314148060049
	-0.00268585193995152	-0.00268585193995149	-0.00268585193995149
24	NAN	NAN	-10^{30}
	NAN	NAN	10^{30}
	NAN	NAN	1
	NAN	NAN	0

Note that in these two cases the exact roots are not provided since the coefficients of the quartics are specified.

other cases. The FQS algorithm at first glance seems also to perform well but in cases 3, 19, and 22 is not able to handle large roots (giving NANs), and in case 12 the relative error is rather large.

The FLO algorithm is surely rather accurate and it seems to handle all cases well, although overall it is a bit less accurate than OQS. Nevertheless, FQS and OQS can easily exhibit opposite behavior by just slightly changing the values of the roots as shown by cases 20 and 21. We note that in all cases where OQS produces exact results and FLO does not, the accuracy of the latter is in any case below machine accuracy. In addition, in case 15, FLO reconstructs the roots exactly, while OQS exhibits a finite relative error. The STR and FQS algorithms do not handle large roots well and STR is not sufficiently accurate in case 4. The HQR algorithm is clearly not on par with OQS and FLO in terms of accuracy, but overall the accuracy is acceptable. HQR exhibits a large error in case of quadruple roots (case 14), which are rather unlikely in practical applications, such as simulation of HCs [43]. HQR also does not handle well the large roots of cases 19 and 22.

Finally, the analytic solution FER, as expected, is rather inaccurate and thus useless for practical applications.

Both the FLO and OQS algorithms employ polynomial rescaling as discussed in Section 2.2; hence, they are able to handle quartics with very large roots as shown in case 22.

Concerning cases 23 and 24, the results are shown in Table 3 for all the quartic solvers. In case 23, all algorithms exhibit a consistent behavior except for STR, which seems not to handle well the fact that $d_2 \approx 0$. Case 24 is more problematic since very widely spread coefficients cause FER and FQS to fail badly and STR to provide two largely wrong roots. The HQR algorithm also calculates the tiniest root wrongly. On the contrary, the OQS and FLO algorithms also confirm their very good accuracy in both of these two extreme cases. Note that in these two cases, the exact roots are not known, since the coefficients of the quartics are specified; anyway, the roots obtained by OQS and FLO are almost identical (i.e., with a maximum relative error ε_{max} less than 10^{-15}) to the ones that can be obtained by the NSolve function of Wolfram Mathematica using a working precision of 16.

Table 4. Analytical Expressions Used to Generate the Roots Used for the Statistical Analysis

Sample	Roots				Note
	x_1	x_2	x_3	x_4	
$\mathcal{A}$	ξ_1	ξ_2	ξ_3	ξ_4	Four small real
$\mathcal{B}$	ξ_1	ξ_2	$\xi_3 + \xi_4 i$	x_3^*	Two small real, two small complex
$\mathcal{C}$	$\xi_1 + \xi_2 i$	x_1^*	$\xi_3 + \xi_4 i$	x_3^*	Four small complex
$\mathcal{D}$	ξ_1	ξ_2	$(\xi_3 + \xi_4 i)\, 10^6$	x_3^*	Two small real, two large complex
$\mathcal{E}$	$(\xi_1 + \xi_2 i)\, 10^6$	x_1^*	$(\xi_3 + \xi_4 i)\, 10^6$	x_3^*	Four large complex
	Quartic Coefficients				
	a	b	c	d	
$\mathcal{F}$	ξ_1	ξ_2	ξ_3	ξ_4	Four random quartic coefficients

$\xi_i \in \mathrm{Re}$ are uniform random variates on $(-0.5, 0.5)$. The $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ samples each consist of 2×10^{11} sets of four roots and each set of four roots is used to generate the quartic coefficients through Equation (109). Sample $\mathcal{F}$ consists of 1.5×10^{11} sets of quartic coefficients.

Finally, we observe that, in principle, some improvements suggested in the methods, such as the shift of x in the resolvent cubic (Section 2.2) or the use of the Newton-Raphson method (Section 2.2), could be regarded as useless. Anyway, by renouncing these improvements, the accuracy of OQS significantly degrades. For example, without using the Newton-Raphson method and the x-shift, the maximum relative error ε_{max} of OQS in cases 2, 12, and 17 becomes 0.029, 0.015, and 0.0002, respectively, i.e., many orders of magnitude larger.

3.2 Statistical Analysis

To mimic a realistic application, we tested the quartic solvers over a very large set of randomly generated quartic equations. We generated randomly six large sets $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}$—which in the following we call samples—of quartic equations as shown in Table 4. In the case of the first five samples from $\mathcal{A}$ to $\mathcal{E}$, the quartic coefficients are calculated using quadruple precision from the randomly generated roots by using Equation (109), while in the case of sample $\mathcal{F}$ the quartic coefficients are randomly generated. The samples $\mathcal{B}$, $\mathcal{D}$, and $\mathcal{E}$ are designed to reproduce what happens in Monte Carlo (MC) simulations of hard cylinders (HCs), where the HC overlap detection is based on the solution of a quartic equation [43].

For the samples $\mathcal{A}$–$\mathcal{E}$ the statistical analysis was carried out by generating a set of 2×10^{11} roots, while sample $\mathcal{F}$ consists of 1.5×10^{11} quartics with randomly generated coefficients. For the corresponding set of quartics we calculated the roots using all the solvers we considered and we estimated the probability distribution $P(\varepsilon_{rel})$ of the relative error ε_{rel} of each calculated root, where given the exact root x_i and the calculated one x_i^c, one has

$$\varepsilon_{rel} = \begin{cases} \frac{|x_i^c - x_i|}{|x_i|} & \text{if } x_i \neq 0 \\ |x_i^c| & \text{otherwise.} \end{cases} \tag{111}$$

Since $P(\varepsilon_{rel})$ is a probability distribution, if one calculates N_{tot} roots, the number dN of them with a relative error between ε_{rel} and $\varepsilon_{rel} + d\varepsilon_{rel}$ (with $d\varepsilon_{rel} \ll \varepsilon_{rel}$) is

$$dN = N_{tot} P(\varepsilon_{rel}) d\varepsilon_{rel}. \tag{112}$$

From $P(\varepsilon_{rel})$ we calculated the cumulative distribution function $F(\varepsilon_{rel})$, i.e.,

$$F(\varepsilon_{rel}) = \int_{\varepsilon_{rel}}^{+\infty} P(x)\, dx, \tag{113}$$

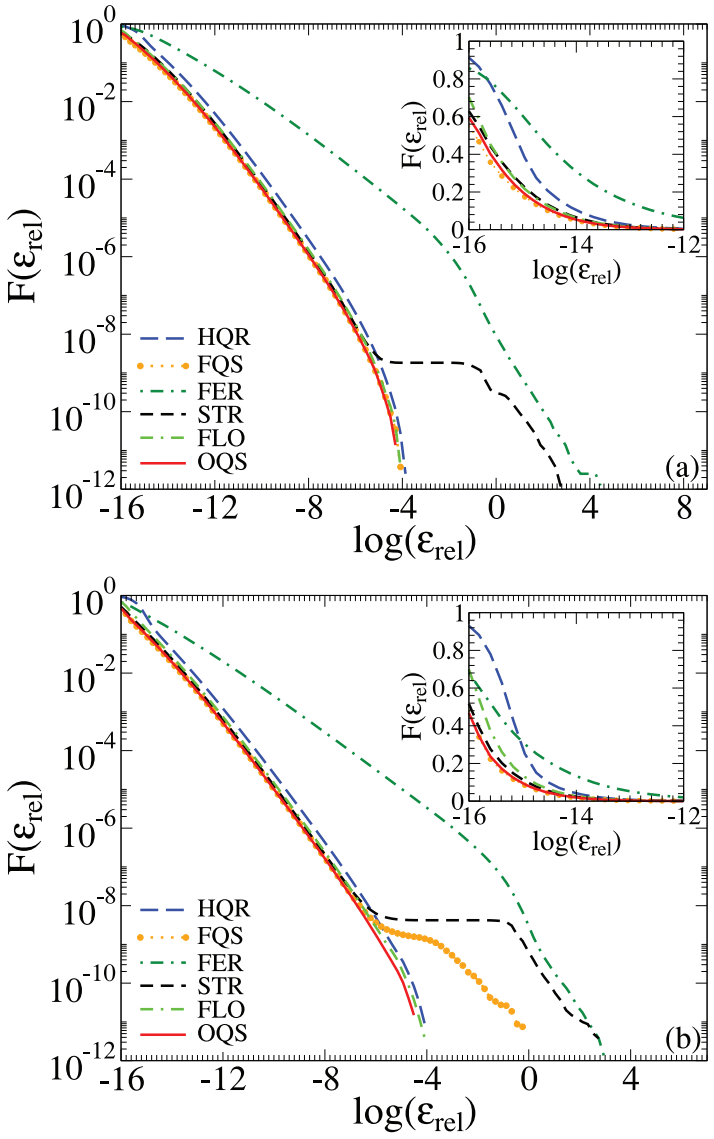

Fig. 2. Plot of the cumulative distribution function $F(\varepsilon_{rel})$ for samples $\mathcal{A}$ (a) and $\mathcal{B}$ (b) for all the algorithms considered in the present article. Insets show the same plots but using a linear scale on y-axis.

which is the probability of having a relative error greater than ε_{rel}. In the case of sample $\mathcal{F}$, the reference roots x_i in Equation (111) have been calculated by using the HQR algorithm in quadruple precision. We chose the *HQR* algorithm to calculate the reference roots since for this specific sample it is expected to be rather reliable and we were able to easily implement it in quadruple precision.

A preliminary observation about the use of the *log-log* scale for plots shown in Figures 2 through 4 is in order. If one uses a *log-lin* plot, as in the insets of these figures, large errors that are quite unlikely cannot be detected, but these errors are the most critical ones for practical applications

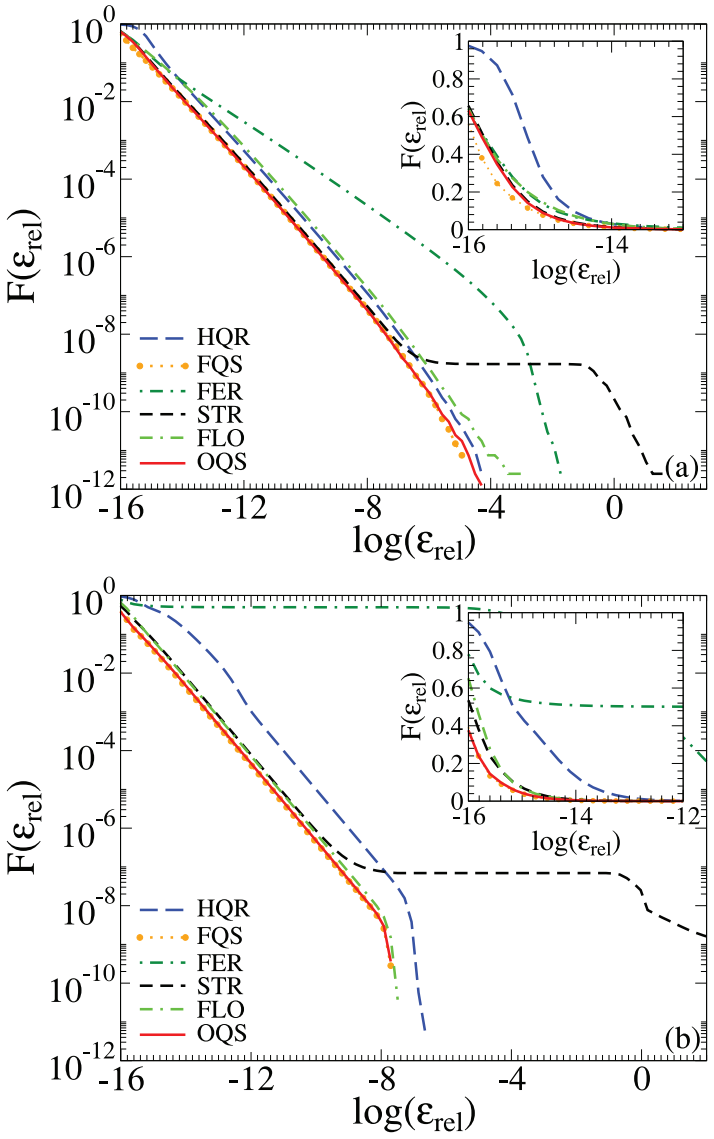

Fig. 3. Plot of the cumulative distribution function $F(\varepsilon_{rel})$ for samples $\mathcal{C}$ (a) and $\mathcal{D}$ (b) for all the algorithms considered in the present article. Insets show same plots but using a linear scale on the y-axis.

where robustness cannot be sacrificed. A first conclusion that can immediately be drawn from the results of our statistical analysis is that again, the FER analytic solution is not able to handle numerical errors properly, since, as it can be seen in Figures 2 through 4, very large errors are very likely for all investigated samples. With sample $\mathcal{F}$, only the algorithms OQS, HQR, and FLO exhibit a consistent behavior since for them the largest observed relative error is smaller than about 10^{-8}. In passing, we note that solutions of sample $\mathcal{F}$ are more accurate than the ones of the other samples. The reason for this result can be rooted in the fact that for this sample, closely spaced roots, which are the ones that produce solutions with larger errors, are much less likely. With this sample the algorithms HQR and OQS provide almost the same accuracy, while FLO

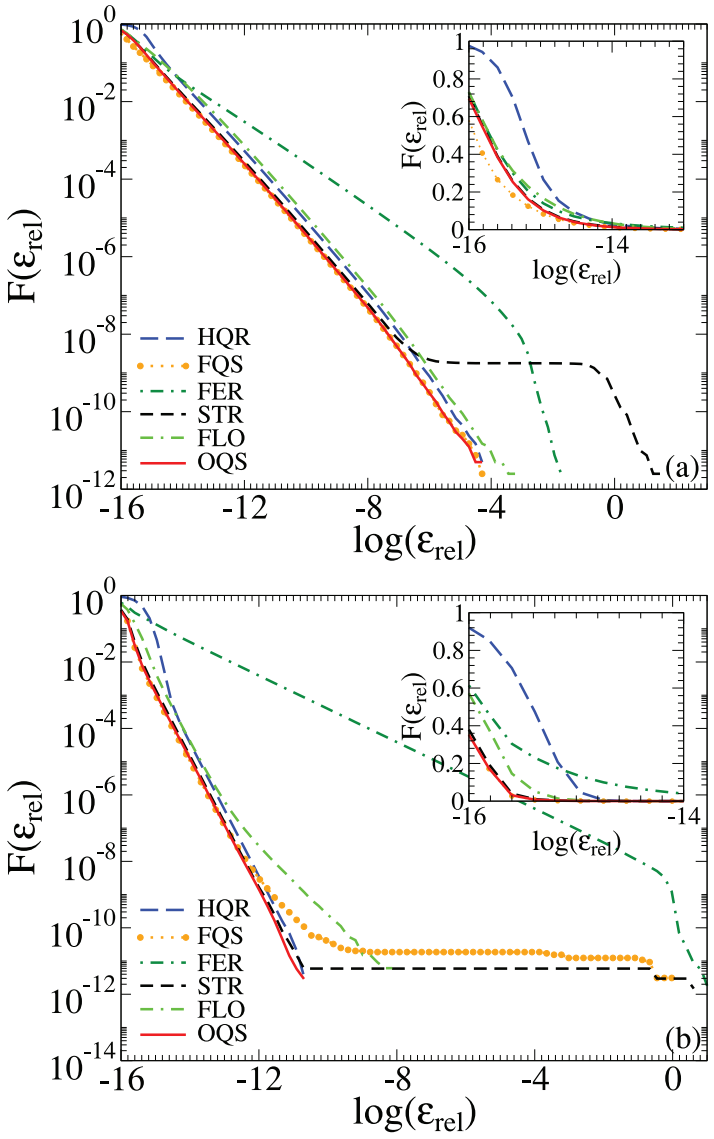

Fig. 4. Plot of the cumulative distribution function $F(\varepsilon_{rel})$ for samples $\mathcal{E}$ (a) and $\mathcal{F}$ (b) for all the algorithms considered in the present article. Insets show same plots but using a linear scale on the y-axis.

shows a clear deviation in the tail of the distribution, which means that in this case, for the latter algorithm large errors are a little more likely. The STR algorithm has some serious issues with all samples since its $F(\varepsilon_{rel})$ below about 10^{-8} exhibits a very large shoulder (see Figures 2 through -4). A similar behavior can be observed for the FQS algorithm with samples $\mathcal{B}$ and $\mathcal{F}$, which are shown in Figures 2(b) and 4(b), respectively. These shoulders in the tail of $F(\varepsilon_{rel})$ mean that for the STR and FQS algorithms, very large errors are quite likely, thus questioning their use in practical applications. For example, in MC simulation of 1,000 HCs [43] at moderate concentrations lasting 5×10^7 steps, one could expect to calculate the roots of 5×10^{11} quartics, thus being very likely to stumble on unacceptably large errors.

Table 5. CPU Timings $\tau_{exe}/10^8$ (Expressed in ns) with the Associated Absolute Error for All the Tested Quartic Solvers Obtained as Averages over 10 Independent Runs

Opt. Level	Quartic Equation Solver					
	OQS	FLO	STR	FER	FQS	HQR
compiler: apple-gcc						
-O3	236 ± 1	579 ± 3	366 ± 3	154 ± 2	562 ± 1	1132 ± 4
-O0	476 ± 4	887 ± 2	589 ± 2	172 ± 1	1112 ± 6	2343 ± 3
compiler: gnu-gcc						
-O3	252 ± 1	561 ± 2	384 ± 4	140 ± 1	580 ± 1	1028 ± 1
-O0	491 ± 1	902 ± 7	631 ± 7	177 ± 1	1168 ± 2	2577 ± 2

During each run, a set of 10^8 quartic polynomials from sample $\mathcal{B}$ is generated. We performed these timing tests using both apple-gcc and gnu-gcc with (-O3) and without (-O0) optimizations.

Overall, OQS, FLO, and HQR seem to handle numerical errors well, with OQS more accurate than the other two though. The gap in accuracy between OQS and HQR is particularly evident with sample $\mathcal{D}$, as shown in Figure 3(b). The FLO algorithm is fairly accurate with samples $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$, but its accuracy degrades slightly with samples $\mathcal{C}$, $\mathcal{E}$, and, more significantly, $\mathcal{F}$ (see Figures 3(a), 4(a), and 4(b), respectively).

3.3 Timing Test

Our last test concerns the efficiency of the quartic equation solvers. We generated 10 sets of 10^8 quartics as we did for sample $\mathcal{B}$ in Section 3.2 for each quartic solver (see Table 4) and we recorded the total execution time τ_{tot} for each set. The execution time τ_{exe} of each solver is thus $\tau_{exe} = \tau_{tot} - \tau_0$, where τ_0 is the execution time of a run during which we do not solve the quartic equation but we only generate random roots and calculate the corresponding quartic coefficients. The resulting timings τ_{exe} of all the algorithms—averaged over the 10 independent realizations—are shown in Table 5 together with the associated absolute errors.

To check the dependency on compiler type and optimization level of our results, we recorded timings with very high (-O3) and without (-O0) compiler optimizations turned on and we also used two different compilers to generate the code for these tests (apple-gcc and gnu-gcc). For this sample, it can be seen that OQS is almost on par with the analytic solution; it is about twice as fast as the FLO algorithm and almost 5 times faster than the HQR solver. The STR algorithm also performs quite well, being only 1.7 times slower than OQS, but we already noted that it is not accurate enough and robust enough for practical applications.

We note that sample $\mathcal{B}$ has been chosen because this set of roots are similar to the ones that can be obtained in [31]. Anyway, timings may depend on the sample used for the test, and hence we also report in Table 6 timings for sample $\mathcal{F}$, which includes all forms of quartics of the samples $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ combined. With this sample the FLO and STR algorithms perform better and their timings are closer to the ones obtained by the OQS algorithm, while the results for the other algorithms do not change much. The FLO algorithm is generally slower than OQS, due to its additional time-consuming coded features like polynomial rescaling and reordering of the resulting roots.

Despite the HQR algorithm being much slower than the other ones, we remind the reader that it can be used for polynomials of arbitrary degree. Compared to other general purpose algorithms, such as MPSolve [8] (MPS) or the one proposed in [6] (AUR), it proved to be more efficient. We carried out some timing tests based on sample $\mathcal{B}$ and $\mathcal{F}$ for MPS and AUR (in its double shift variant) methods, whose source codes are freely available. The MPS and AUR algorithms turned out to be around 80 and 5 times slower than HQR, respectively. Performance benchmarks of AUR

Table 6. Same as for Table 5 but for Sample $\mathcal{F}$

Opt. Level	Quartic Equation Solver					
	OQS	FLO	STR	FER	FQS	HQR
compiler: apple-gcc						
-O3	249 ± 1	481 ± 1	324 ± 1	167 ± 1	551 ± 1	1118 ± 7
-O0	467 ± 4	741 ± 7	524 ± 6	182 ± 3	1036 ± 12	2443 ± 16
compiler: gnu-gcc						
-O3	251 ± 2	462 ± 8	363 ± 8	156 ± 2	521 ± 6	1031 ± 2
-O0	490 ± 10	771 ± 9	554 ± 2	186 ± 3	1050 ± 6	2620 ± 20

can also be found in [6] where, for polynomials with degree smaller than 10, the HQR algorithm turns out to be about 5 times faster than AUR, thus confirming our results.

4 CONCLUSIONS

In this article, we propose a novel algorithm to solve a quartic equation based on the factorization of the quartic polynomial into two quadratic factors and the use of the NR method to refine the coefficients of quadratic polynomials.

Our method can be regarded as an analytic approach where all the calculations have been carefully done to alleviate the detrimental effect of cancellation errors. In this respect, there are attempts to improve analytic methods by minimizing the effect of numerical errors as discussed in [27], and it would be interesting to compare them against the present algorithm, which could be a matter for a future publication.

Our solver is very accurate and robust according to our accuracy tests for a selected set of extreme cases and to our statistical tests, which have been carried out over a very large sample of quartics (up to 2×10^{11}). Indeed, in comparison with the FLO and HQR algorithms, which are the only viable alternatives in terms of accuracy and robustness, our solver performs better.

Our solver also proved to be very efficient, being almost on par with the analytic solutions, about twice as fast as the FLO algorithm, and almost 5 times faster than the HQR algorithm. These features of our solver make it very suitable for applications where robustness, accuracy, and efficiency are all crucial.

In soft matter physics, coarse-grained models of molecules based on hard rigid bodies have been commonly used over last years to study colloidal systems, i.e., water suspensions of particles, that have a diameter of between approximately 1 and 1,000nm [19, 47, 48, 54, 58]. Models based on HCs have been widely employed to study the physical properties of colloidal systems made of rod-like particles [16, 17, 34, 35, 42], and simulations of HCs are rather demanding in terms of both accuracy and efficiency, as discussed in [43].

In the latter article a novel algorithm for testing the overlap of two HCs, which is based on the solution of the quartic equation, has been proposed. Since the check for the overlap of two hard spherocylinders (HSCs) [3], intended as an alternative model for rod-like colloidal particles, can be performed very efficiently [56], in the past they have been preferred to HCs for carrying out computer simulations [10, 37, 55]. Anyway, thanks to the efficiency of our quartic equation solver, simulations of HCs are almost as efficient as those of HSCs as shown in [43]. In [43], a stringent test of robustness of OQS is also provided. An MC simulation of 980 HCs has been carried out, which lasted 5×10^7 MC steps, and the resulting HC trajectories turned out to be identical to the ones obtained by two other tested algorithms (see Figure 5 in [43]). Finally, our algorithm has also been generalized to solve quartic polynomials with complex coefficients.

In conclusion, due to its robustness, accuracy, and efficiency, we are confident that our quartic equation solver will be a very valuable tool for a wide spectrum of scientific applications ranging from physics, chemistry, and mathematics to engineering and computer science.

REFERENCES

[1] M. Abramowitz and I. A. Stegun. 1972. *Handbook of Mathematical Functions* (10th ed.). National Bureau of Standards, New York.

[2] A. C. Aitken. 1952. XXIII.—Studies in practical mathematics. VII. On the theory of methods of factorizing polynomials by iterated division. In *Proceedings of the Royal Society of Edinburgh. Section A. Mathematical and Physical Sciences*, Vol. 63. Royal Society of Edinburgh, Edinburgh, 326–335. DOI: https://doi.org/10.1017/S0080454100007202

[3] M. P. Allen, G. T. Evans, D. Frenkel, and B. M. Mulder. 1993. *Hard Convex Body Fluids*. John Wiley & Sons, Inc., New York, 1–166. DOI: https://doi.org/10.1002/9780470141458.ch1

[4] R. Alt and J. Vignes. 1982. Stabilizing Bairstow's method. *Comput. Math. Appl.* 8, 5 (1982), 379–387.

[5] D. W. Arthur. 1972. Extension of Bairstow's method for multiple quadratic factors. *IMA J. Appl. Math.* 9, 2 (1972), 194–197.

[6] J. Aurentz, T. Mach, R. Vandebril, and D. Watkins. 2015. Fast and backward stable computation of roots of polynomials. *SIAM J. Matrix Anal. Appl.* 36, 3 (2015), 942–973.

[7] L. Bairstow. 1914. *Investigations Relating to the Stability of the Aeroplane*. Reports and Memoranda 154. National Advisory Committee for Aeronautics.

[8] D. A. Bini and G. Fiorentino. 2000. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numer. Algorithms* 23, 2 (June 2000), 127–173.

[9] G. M. Birtwistle and D. J. Evans. 1967. On the generalisation of Bairstow's method. *BIT Numer. Math.* 7, 3 (1967), 175–190.

[10] P. Bolhuis and D. Frenkel. 1997. Tracing the phase boundaries of hard spherocylinders. *J. Chem. Phys.* 106, 2 (1997), 666–687.

[11] K. M. Borkowski. 1987. Transformation of geocentric to geodetic coordinates without approximations. *Astrophys. Space Sci.* 139, 1 (1987), 1–4.

[12] C. B. Boyer and U. C. Merzbach. 1991. *A History of Mathematics* (2nd ed.). Wiley, New York.

[13] K. W. Brodlie. 1975. On Bairstow's method for the solution of polynomial equations. *Math. Comp.* 29, 131 (1975), 816–826.

[14] F. M. Carrano. 1973. A modified Bairstow method for multiple zeros of a polynomial. *Math. Comp.* 27, 124 (1973), 781–792.

[15] B. Christianson. 1991. Solving quartics using palindromes. *Math. Gaz.* 75, 473 (1991), 327–328.

[16] C. De Michele, T. Bellini, and F. Sciortino. 2012. Self-assembly of bifunctional patchy particles with anisotropic shape into polymers chains: Theory, simulations, and experiments. *Macromolecules* 45, 2 (2012), 1090–1106. DOI: https://doi.org/10.1021/ma201962x

[17] C. De Michele, L. Rovigatti, T. Bellini, and F. Sciortino. 2012. Self-assembly of short DNA duplexes: From a coarse-grained model to experiments through a theoretical link. *Soft Matter* 8, 32 (2012), 8388–8398. DOI: https://doi.org/10.1039/C2SM25845E

[18] J. Dexter and E. Agol. 2009. A fast new public code for computing photon orbits in a Kerr spacetime. *Astrophys. J.* 696, 2 (2009), 1616.

[19] J. P. K. Doye, A. A. Louis, I-C. Lin, L. R. Allen, E. G. Noya, A. W. Wilber, H. C. Kok, and R. Lyus. 2007. Controlling crystallization and its absence: Proteins, colloids and patchy models. *Phys. Chem. Chem. Phys.* 9, 18 (2007), 2197–2205.

[20] C. D'Souza. 1997. *An Optimal Guidance Law for Planetary Landing*. American Institute of Aeronautics and Astronautics, Reston, VA, 1376–1381. DOI: https://doi.org/10.2514/6.1997-3709

[21] T. F. W. Embleton, G. J. Thiessen, and J. E. Piercy. 1976. Propagation in an inversion and reflections at the ground. *J. Acoust. Soc. Am.* 59, 2 (1976), 278–282.

[22] B. T. Fang. 1990. Simple solutions for hyperbolic and related position fixes. *IEEE Trans. Aerosp. Electron. Syst.* 26, 5 (1990), 748–753.

[23] T. Fiala and A. Krebsz. 1986. On the convergence and divergence of Bairstow's method. *Numer. Math.* 50, 4 (1986), 477–482.

[24] N. Flocke. 2015. Algorithm 954: An accurate and efficient cubic and quartic equation solver for physical applications. *ACM Trans. Math. Softw.* 41 (2015), 30:1–30:24.

[25] G. H. Golub and C. F. Van Loan. 1990. *Matrix Computations* (2nd ed.). John Hopkins University Press, Baltimore, MD.

[26] A. A. Grau. 1960. Solution of polynomial equation by Bairstow-Hitchcock method. *Commun. ACM* 3, 2 (1960), 74–75.

[27] Don Herbison-Evans. 1995. Solving quartics and cubics for graphics. In *Graphics Gems V*. Academic Press, 3–15.

[28] M. Igarashi. 1984. A termination criterion for iterative methods used to find the zeros of polynomials. *Math. Comp.* 42, 165 (1984), 165–171.
[29] V. Pan J. M. McNamee. 2013. *Numerical Methods for Roots of Polynomials - Part II* (1st ed.). Vol. 16. Elsevier.
[30] M. A. Jenkins. 1975. Algorithm 493: Zeros of a real polynomial [C2]. *ACM Trans. Math. Softw.* 1, 2 (1975), 178–189.
[31] M. A. Jenkins and J. F. Traub. 1970. A three-stage algorithm for real polynomials using quadratic iteration. *SIAM J. Numer. Anal.* 7, 4 (1970), 545–566.
[32] T. B. Kaiser. 2000. Laser ray tracing and power deposition on an unstructured three-dimensional grid. *Phys. Rev. E* 61 (2000), 895–905.
[33] G. A. Korn and T. M. Korn. 2000. *Mathematical Handbook for Scientists and Engineers*. Dover Publications.
[34] T. Kouriabova, M. D. Betterton, and M. A. Glaser. 2010. Linear aggregation and liquid-crystalline order: Comparison of Monte Carlo simulation and analytic theory. *J. Mater. Chem.* 20 (2010), 10366–10383.
[35] X. Lü and J. T. Kindt. 2004. Monte Carlo simulation of the self-assembly and phase behavior of semiflexible equilibrium polymers. *J. Chem. Phys.* 120 (2004), 10328–10338.
[36] X. Luo, Q. Wang, C. Yang, and F. Liu. 2006. Detection of LTSB steganography based on quartic equation. In *2006 8th International Conference Advanced Communication Technology*, Vol. 2. 1199–1204. DOI: https://doi.org/10.1109/ICACT.2006.206186
[37] S. C. McGrother, D. C. Williamson, and G. Jackson. 1996. A re–examination of the phase diagram of hard spherocylinders. *J. Chem. Phys.* 104, 17 (1996), 6755–6771.
[38] J. M. McNamee. 1993. A bibliography on roots of polynomials. *J. Comput. Appl. Math.* 47, 3 (1993), 391–394.
[39] J. M. McNamee. 1999. An updated supplementary bibliography on roots of polynomials. *J. Comput. Appl. Math.* 110, 2 (1999), 305–306.
[40] O. Naroditsky, A. Patterson, and K. Daniilidis. 2011. Automatic alignment of a camera with a line scan LIDAR system. In *2011 IEEE International Conference on Robotics and Automation*. 3429–3434. DOI: https://doi.org/10.1109/ICRA.2011.5980513
[41] S. Neumark. 1965. *Solution of Cubic and Quartic Equations* (1st ed.). Pergamon Press, Oxford.
[42] K. T. Nguyen, F. Sciortino, and C. De Michele. 2014. Self-assembly-driven nematization. *Langmuir* 30 (2014), 4814–4819. DOI: https://doi.org/10.1021/la500127n
[43] A. G. Orellana, E. Romani, and C. De Michele. 2018. Speeding up Monte Carlo simulation of patchy hard cylinders. *Eur. Phys. J. E* 41 (2018), 1–10. DOI: https://doi.org/10.1140/epje/i2018-11657-0
[44] M. Phatak, M. Chansarkar, and S. Kohli. 1999. Position fix from three GPS satellites and altitude: A direct method. *IEEE Trans. Aerosp. Electron. Syst.* 35, 1 (1999), 350–354.
[45] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes - The Art of Scientific Computing* (3rd ed.). Cambridge University Press, Cambridge, UK.
[46] H. E. Salzer. 1960. A note on the solution of quartic equations. *Math. Comput.* 14, 71 (1960), 279–281.
[47] F. Sciortino. 2010. Primitive models of patchy colloidal particles. A review. *Collect. Czech. Chem. Commun.* 75 (2010), 349–358.
[48] F. Sciortino and E. Zaccarelli. 2017. Equilibrium gels of limited valence colloids. *Curr. Opin. Colloid Interface Sci.* 30 (2017), 90–96.
[49] S. L. Shmakov. 2011. A universal method of solving quartic equations. *Int. J. Pure Appl. Math.* 71, 2 (2011), 251–259.
[50] L. Sonnenschein. 2006. Analytical solution of $t\bar{t}$ dilepton equations. *Phys. Rev. D* 73, 5 (2006), 054015.
[51] M. Vander Stracten and H. Van de Vel. 1992. Multiple root-finding methods. *J. Comput. Appl. Math.* 40, 1 (1992), 105–114.
[52] P. Strobach. 2010. The fast quartic solver. *J. Comput. Appl. Math.* 234, 10 (2010), 3007–3024.
[53] P. Strobach. 2015. The Low-Rank LDLT Quartic Solver. (2015). DOI: https://doi.org/10.13140/2.1.3955.7440 AST-Consulting Inc. *Internal Technical Report.*
[54] P. I. C. Teixeira and J. M. Tavares. 2017. Phase behaviour of pure and mixed patchy colloids – theory and simulation. *Curr. Opin. Colloid & Interface Sci.* 30 (2017), 16–24.
[55] J. A. C. Veerman and D. Frenkel. 1991. Relative stability of columnar and crystalline phases in a system of parallel hard spherocylinders. *Phys. Rev. A* 43, 8 (1991), 4334–4343.
[56] C. Vega and S. Lago. 1994. A fast algorithm to evaluate the shortest distance between rods. *Comput. Chem.* 18, 1 (1994), 55–59.
[57] M. D. Yacoub and G. Fraidenraich. 2012. A solution to the quartic equation. *Math. Gaz.* 96, 536 (2012), 271–275.
[58] Gi-Ra Yi, David J. Pine, and Stefano Sacanna. 2013. Recent progress on patchy colloids and their self-assembly. *J. Phys: Condens. Matter* 25, 19 (2013), 193101.
[59] W. Zhang, D. Duan, and L. Yang. 2009. Relay selection from a battery energy efficiency perspective. In *2009 IEEE Military Communications Conference (MILCOM'09)*. 1–7. DOI: https://doi.org/10.1109/TCOMM.2011.041111.100128

[60] X. Zheng and P. Palffy-Muhoray. 2007. Distance of closest approach of two arbitrary hard ellipses in two dimensions. *Phys. Rev. E* 75, 6 (2007), 061709.

[61] J. Zhu. 1994. Conversion of earth-centered earth-fixed coordinates to geodetic coordinates. *IEEE Trans. Aerosp. Electron. Syst.* 30, 3 (1994), 957–961.

Received December 2018; revised February 2020; accepted February 2020