Remark on Algorithm 1010: Boosting efficiency in solving quartic equations with no compromise in accuracy.

CRISTIANO DE MICHELE, Dipartimento di Fisica, "Sapienza" Università di Roma, Italy

In this second remark we present a revised correction to Algorithm 1010 [A. Orellana and C. De Michele 2020] with respect to the one already proposed in the remark on Algorithm 1010 [C. De Michele 2022]

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

This remark details a source change that is required to the software package associated with [3] to fix bad estimates of roots for special fourth-degree polynomials.

The present algorithm is based on LDL^t decomposition and if d_2, as defined in [3, (65)], is close to 0 a special strategy has to be used to calculate quartic roots, since the LDL^t decomposition cannot be achieved in this case. In a previous remark [1] we attempted to correct the problem by making this condition less stringent. However in some special cases the new condition proposed in [1, (2)] is still not sufficient. For example, if we consider special fourth-degree polynomials of the form

$$x^4 + Ax^2 + B = 0 \tag{1}$$

then for some choices of the coefficients A and B the present algorithm provides bad estimates of the roots. For example, the exact roots of the following polynomial

$$x^4 - 11x^2 - 46 = 0 \tag{2}$$

are

$$\pm i\sqrt{\frac{1}{2}\left(\sqrt{305} - 11\right)}, \pm\sqrt{\frac{1}{2}\left(11 + \sqrt{305}\right)} \tag{3}$$

but the code provided in the first remark, [1], yields the following incorrect roots:

$$\pm 2.345207900985139, \pm 2.345207858838290 \tag{4}$$

We could consider these polynomials as special cases and solve them by setting $t = x^2$, thus reducing the problem to the numerical solution of a quadratic equation. Nevertheless, the cause of the problem is the same as the one discussed in the previous remark [1], i.e., in these cases the best numerical solution is the one which can be obtained by assuming $d_2 = 0$ (see case 3 in Sec. 2 of [3]), but the condition [1, (2)] is not fulfilled for some

Author's address: Cristiano De Michele, Dipartimento di Fisica, "Sapienza" Università di Roma, P.le A. Moro, 2, Rome, I-00185, Italy, cristiano.demichele@uniroma1.it.

ACM 1557-7295/2024/6-ART
https://doi.org/10.1145/3674833

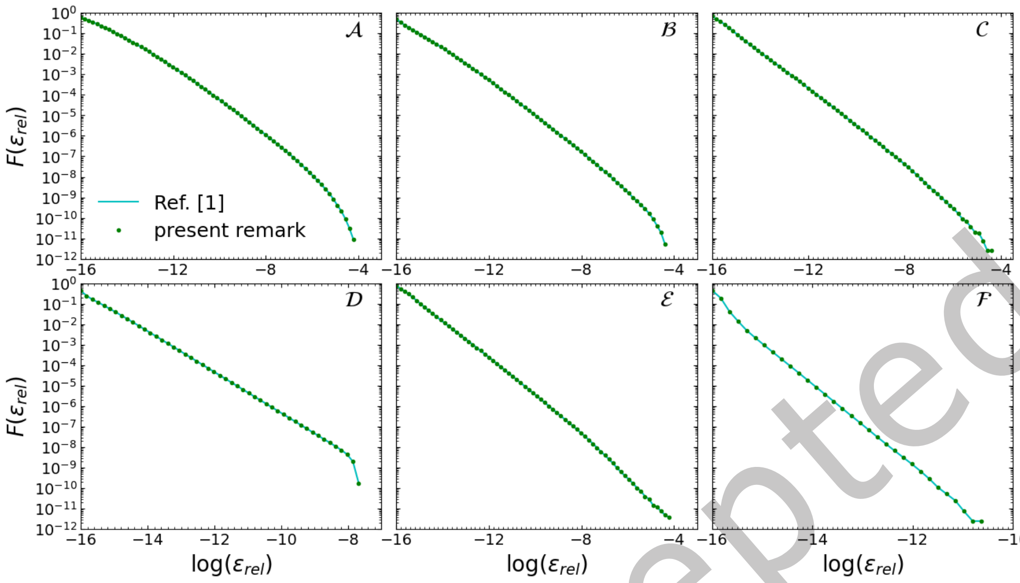

Fig. 1. Plot of the cumulative distribution function $F(\varepsilon_{rel})$ for samples $\mathcal{A} - \mathcal{F}$ where straight lines are results from original algorithm and symbols are ones from codes with changes discussed in the present remark paper.

polynomials of the form (1). Therefore, we have opted to seek a general solution for these polynomials instead of treating them as special cases whose roots can be obtained by solving a quadratic equation.

We note that, if the condition in [3, (65)] is fulfilled, the accuracy of the roots obtained in the case $d_2 = 0$ is compared against the one of the roots obtained by assuming $d_2 \neq 0$. As a consequence, a possible effective fix consists in simply removing this condition in [3, (65)], thus always comparing the root obtained by assuming $d_2 = 0$ with those obtained for $d_2 \neq 0$. However in this case algorithm efficiency degrades by around 10%. Hence, the strategy, which we decided to adopt, is to significantly weaken the condition in [1, (2)], i.e., we employ the following new condition:

$$d_2 \leq \epsilon_c \left(|2b/3| + |\phi_0| + l_1^2\right) \tag{5}$$

where ϵ_c is a tunable parameter (see below), which we set to 1.490116×10^{-8} by default. Note that in [1, (2)] we set ϵ_c equal to the *machine epsilon* ϵ_m, i.e we set $\epsilon_c = \epsilon_m \approx 2.22045 \times 10^{-16}$ (in double precision). As a result of this fix, we have added the polynomial given in (2) to our accuracy tests as case 26.

In addition, as illustrated in the source files `Drivers/simpletest.c` and discussed in the `USER_MANUAL` (included in the source package) the value of the parameter ϵ_c can be tuned, if needed, and one can even force the calculation of roots for the case $d_2 = 0$, thus ignoring the condition in (5) (i.e., as if $\epsilon_c = \infty$)

This change to the algorithm does not affect the accuracy tests as reported in [3, Table 1] at all. In addition, this change has no impact on algorithm efficiency as measured by the timing tests bundled with the present remark software.

In Fig. 1 we show the curves of the statistical analysis, as discussed in the original paper (see [3, Section 3.2]), obtained with both the original code and the new code. The present statistical analysis was carried out by generating a total of 2×10^{11} quartic polynomials for all samples and it may be seen that the cumulative probabilities in all samples $\mathcal{A}$-$\mathcal{F}$ are identical to originals. We finally note that the same change has also been applied to the version of the quartic solver for polynomials with complex coefficients.

In addition to the sources and drivers which accompany the present remark, an updated C++ implementation of the algorithm is freely available (see [2]).

ACKNOWLEDGMENTS

The work is supported by the European Union — Next Generation EU under Grant PRIN2022—TAMeQUAD (No. B53D23004500006)

REFERENCES

[1] Cristiano De Michele. 2022. Remark on Algorithm 1010: Boosting Efficiency in Solving Quartic Equations with No Compromise in Accuracy. *ACM Trans. Math. Softw.* 48, 4, Article 46 (Dec 2022), 3 pages. https://doi.org/10.1145/3564270

[2] C. De Michele. 2020. *quarticpp*. github. http://github.com/cridemichel/quarticpp

[3] 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

Received 21 March 2024; revised 18 April 2024; accepted 20 May 2024