Computing at Least One of Two Roots of a Polynomial is, in General, not Algorithmic

Vladik Kreinovich
Department of Computer Science
University of Texas, El Paso, TX 79968
vladik@utep.edu

Abstract

In our previous work, we provided a theoretical explanation for an empirical fact that it is easier to find a unique root than the multiple roots. In this short note, we strengthen that explanation by showing that finding one of many roots is also difficult.

Background. The main objective of interval mathematics is to produce guaranteed (verified) results. Before we start designing an algorithm that would produce such results for some class of problems, it is desirable to know whether a general algorithm is indeed possible for this class.

Among these problems are solving systems of equations (in particular, equations), finding optima of a given function on a given box, etc.

It is known that there exists an algorithm which is applicable to every system $f_1(x_1, \ldots, x_n) = 0, \ldots, f_m(x_1, \ldots, x_n) = 0$ with computable functions f_i which has exactly one solution on a given box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$, and which computes this solution – i.e., produces, for every $\varepsilon > 0$, an ε-approximation to this solution [5, 6, 7].

It is also known that no algorithm is possible which is applicable to every computable system which has exactly two solutions and which would return *both* solutions [7]. The proof shows that such an algorithm is not possible even for computable polynomial equations.

Mathematical comment. This algorithmic impossibility is due to the fact that we allow computable polynomials; for polynomials with rational (or even algebraic) coefficients, solution problems are algorithmically decidable; see, e.g., [1, 7, 9, 10].

Practical comment. This result is in good accordance with the empirical fact that in general, it is easier to find a point $(x_1, \ldots, x_n)$, in which a given system of equations has a unique solution than when this system has several solutions; see, e.g., [4].

Formulation of the problem. A natural question is: since in the general two-roots case, we cannot return *both* roots, maybe we can return at least one of them?

Comment. This is actually possible in the example from [7].

What was known. It is known that no such algorithm is possible for *general* computable functions [5, 6]. This construction requires a computable function which is more complex than a polynomial.

What we plan to do. In this paper, we show that already for computable *polynomial* equations, it is impossible to compute even one of the roots.

In this proof, we will use the polynomials with the smallest possible number of variables and of the smallest possible degree. We also prove a similar result for optimization problems.

Definitions. In order to precisely formulate this result, we need to recall the definition of a computable number. Crudely speaking, a real number is computable if it can be computed with an arbitrary accuracy.

Definition 1. (see, e.g., [3, 7]) *A real number x is called* computable *if there exists an algorithm that, given an integer k, returns a rational number r for which $|x - r_k| \le 2^{-k}$.*

By a *computable polynomial*, we mean a polynomial with computable coefficients.

According to our promise, we will prove this result for the case of the smallest possible number of variables: one.

Proposition 1. *No algorithm is possible that is applicable to any computable polynomial function $f(x)$ with exactly two roots, and returns one of these roots.*

Proof. Our proof will use the known fact that no algorithm is possible for detecting whether a given constructive real number α is non-negative or non-positive [3, 8].

For every computable real number α, we can form a polynomial $f_\alpha(x) = [(x-1)^2 + \alpha] \cdot [(x+1)^2 - \alpha]$. This polynomial is equal to 0 if one of the two factors is equal to 0.

- When $\alpha = 0$, this polynomial $f_\alpha(x)$ has exactly two roots: 1 and -1.
- When $\alpha > 0$, the first factor is positive, so $f_\alpha(x) = 0$ if and only if $(x+1)^2 = \alpha$, hence $x + 1 = \pm\sqrt{\alpha}$. So, for such α, the polynomial $f_\alpha(x)$ has exactly two roots $x = -1 \pm \sqrt{\alpha}$.

- Similarly, when $\alpha < 0$, the polynomial $f_\alpha(x)$ has exactly two roots $x = 1 \pm \sqrt{|\alpha|}$.

If we could compute one of roots, then by computing this root with enough accuracy and comparing it with 1, we could tell whether this root is close to 1 or to -1. According to our description of the roots, if this root is close to 1, then $\alpha \le 0$; if this root is close to -1, then $\alpha \le 0$. Since, as we have mentioned, we cannot check whether $\alpha \ge 0$ or $\alpha \le 0$, we thus cannot return one of the roots. The proposition is proven.

Comment. In the proof, we used a 4th degree polynomial. Let us give reasons why we cannot use a polynomial of a lower degree. Since we need polynomials with two roots, we must use polynomials of degree at least 2. If a quadratic polynomial has exactly 2 roots, then we can find these rules by using a standard formula, so these roots are easy to compute.

For a cubic polynomial $f(x)$, the only way to have exactly two real roots is to have one double root. At this root, the derivative $f'(x)$ is equal to 0 – and the solution to the quadratic equation $f'(x) = 0$ can be easily found.

Proposition 2. *No algorithm is possible which is applicable to any computable polynomial $f(x)$ that attains its minimum at exactly two points, and returns one of these points.*

Proof. It is sufficient to consider $f_\alpha^2(x)$, where $f_\alpha(x)$ is the polynomial from the previous proof; this new polynomial is always non-negative, and it attains its minimum 0 if and only $f_\alpha(x) = 0$.

Acknowledgments. This work was supported in part by the Max Planck Institut für Mathematik, by the NSF grants EAR-0225670 and EIA-0080940, by Texas Department of Transportation grant No. 0-5453, and by the Japan Advanced Institute of Science and Technology (JAIST) International Joint Research Grant 2006-08.

The author is thankful to Michael Beeson for the formulation of the problem and stimulating advice, and to the participants of the Conference on the Methods of Proof Theory in Mathematics, Bonn, June 4–10, 2007, for valuable discussions.

References

[1] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in real algebraic geometry*, Springer-Verlag, Berlin, 2006.

[2] M. Beeson, "Some relations between classical and constructive mathematics", *Journal of Symbolic Logic*, 1978, Vol. 43, pp. 228–246.

[3] M. Beeson, *Foundations of Constructive Mathematics: Metamathematical Studies*, Springer, Berlin/Heidelberg/New York, 1985.

[4] R. B. Kearfott, *Rigorous global search: continuous problems*, Kluwer, Dordrecht, 1996.

[5] V. Kreinovich, "Uniqueness implies algorithmic computability", *Proceedings of the 4th Student Mathematical Conference*, Leningrad University, Leningrad, 1975, pp. 19–21 (in Russian).

[6] V. Kreinovich, *Categories of space-time models*, Ph.D. dissertation, Novosibirsk, Soviet Academy of Sciences, Siberian Branch, Institute of Mathematics, 1979, (in Russian).

[7] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

[8] B. A. Kushner, *Lectures on constructive mathematical analysis*, Amer. Math. Soc., Providence, RI, 1985.

[9] B. Mishra, "Computational real algebraic geometry", in: *Handbook on Discreet and Computational Geometry*, CRC Press, 1997.

[10] A. Tarski, *A decision method for elementary algebra and geometry*, 2nd ed., Berkeley and Los Angeles, 1951.