A Hypothetical way to Extrapolate All-Complex Roots of a Polynomial

Dirk Mittler

January 22, 2019

0.0.1 Context:

The methodology exists, by which at least 1 root of a polynomial of degree (N) greater than 2 can be found by successive approximation, until it is accurate to within a specified margin of error, so that next, Factor Theorem can be applied to derive a polynomial of degree (N-1), for which the methodology can be repeated.

0.0.2 Problem 1:

The range of values of (x) to be tried is not known to begin with, so that naive methods of successive approximation might need to be given hints, as to what range to scan for real-numbered roots.

0.0.3 Problem 2:

Both the parameter (x) and the polynomial function (y), with real coefficients (a_0, a_1, a_2, $\cdots a_N$), could be complex numbers, so that at first glance, successive approximation would need to be 2-dimensional. Additionally, every time the degree of the polynomial is even, the possibility exists that not even one of its roots are real, but the complex roots must be found anyway.

0.0.4 Proposed solution:

Instead of just performing a successive approximation, this problem could be conceived at first, to be based on a methodology of linear extrapolation, by which the two end-points (ya) and (yb) could be plotted according to the parameter (t), so that the extrapolated estimate (xt) follows as a 'blended' result between (xa) and (xb). More precisely, the computations to perform could be:

$\epsilon = 10^{-12}$
$xa_0 = -1,\ xb_0 = e^{\pi i/(N+1)}$
$ya_0 \leftarrow xa_0,\ yb_0 \leftarrow xb_0$
$spread_0 = |(yb_0) - (ya_0)|$
$\left[N \leq 4,\ (spread_0)^N < \epsilon\right] \rightarrow Return\ Error$
$\left[N > 4,\ (spread_0)^{(N-2)} < N^2\epsilon\right] \rightarrow Return\ Error$
$misses_0 = 0$

Begin Loop:

$\epsilon' = \max\left(N\left|xa_n\right|^{(N-1)}\ , \frac{1}{N}\right)\ \epsilon$
$|ya_n| \leq \epsilon' \rightarrow Return\,(xa_n)$
$|yb_n - ya_n| \leq \epsilon' \rightarrow Return\,(xa_n)$

$t_n = \frac{-ya_n}{yb_n - ya_n}$
$xt_n = t_n\, xb_n + (1 - t_n)\, xa_n$
$xa_{(n+1)} = xt_n,\ xb_{(n+1)} = xt_n + i\,\frac{xa_n - xt_n}{2}$
$ya_{(n+1)} \leftarrow xa_{(n+1)}$
$yb_{(n+1)} \leftarrow yb_{(n+1)}$

$\left|ya_{(n+1)}\right| \geq spread_n \rightarrow misses_{(n+1)} = misses_n + 1$
$\left|ya_{(n+1)}\right| < spread_n \rightarrow misses_{(n+1)} = misses_n$
$misses_{(n+1)} > N \rightarrow Return\ Error$
$spread_{(n+1)} = \left|ya_{(n+1)}\right|$

End Loop

- It's assumed that the polynomial has been normalized, so that the coefficient (a_N) of the highest-degree term equals $(+1)$.
- Based entirely on instinct, I would suggest that this system could just be extended, so that all the variables are allowed to be complex numbers, including (t_n), in which case (xb_0) may also be initialized with some imaginary component.
- As soon as $|xb_n - xa_n|$ becomes much smaller than the distance between roots, this algorithm should also be faster than a binary search, because over very small intervals, a curve starts to approximate a straight line.
- Caveat: This algorithm will solve a linear equation $(N = 1)$ within one iteration. But this also means that if given an equation of degree $(N > 1)$, it will compute the first approximation of the root, as if the equation was of the first degree, meaning that (xt_0) will be a real number only. This means that (xa_1), (xb_1), (ya_1) and (yb_1) would also be real numbers only. For that reason, in order to be able to find any complex roots, the twister needs to be included, which I wrote above as $(+i\,\frac{xa_n - xt_n}{2})$.
- If the polynomial is non-linear, $(N > 1)$, then it will follow that the complex value (t_n) will not be congruent with $(xb_n - xa_n)$, as (t_n) follows from $(yb_n - ya_n)$, which is equally not congruent. Therefore, the condition cited in the caveat above, might only occur when $(N = 1)$. Yet, without the twister, successive iterations may still 'flatten' the series (xa_n), such that the ability to generate complex values may diminish.

There is a note about finding complex roots which needs to be observed, as providing a context for the algorithm above to be called in. For the sake of argument, the polynomial could be of the 3rd degree, so that being of an odd degree, it will have at least one real root, plus two additional roots, which may or may not be complex. The occurrence of double roots will be ignored for the moment. On the supposition that two of the roots are indeed complex, the algorithm could find one complex root first by chance. The problem which might take place at first glance, is that this complex root could get factorized out of the initial polynomial, so that a quadratic would result, with one real root and one complex root.

The only way in which such a quadratic is possible, is if the coefficients of polynomials are allowed to be complex in turn, and attempting to divide a real-coefficient polynomial by one complex factor, will in fact result in such a polynomial of degree (N-1), that has complex coefficients.

One of the standards which I'm holding this exercise to, is that all coefficients of all polynomials, will be real coefficients.

Therefore, the following idea must be applied, as soon as the algorithm above finds a complex root:

- The conjugate of that root must also be a root,
- These two conjugate roots can be used by conventional means, to derive a quadratic with only real coefficients.
- The polynomial of degree (N) must be divided by that quadratic, in order to arrive at a polynomial of degree (N-2), with only real coefficients.

(b follows from the real component, of the complex root:)

$a = 1$
$xr = \frac{-b}{2a}$
$2a\,(xr) = -b$
$b = -2a\,(xr)$

$xi^2 = \frac{-(b^2-4ac)}{4a^2}$

$xi^2 = \frac{4ac-b^2}{4a^2}$

$4a^2\,(xi)^2 = 4ac - b^2$
$4ac = 4a^2\,(xi)^2 + b^2$

$c = a\,(xi)^2 + \frac{b^2}{4a}$

Dirk Mittler