On Modifications to Laguerre's Method and the Polynomial Eigenvalue Problem

Thomas R. Cameron* & Nikolas I. Steckley**

Davidson College* & DiscoverOrg LLC.**

Abstract

Laguerre's method has long been recognized for its strong virtues when computing the roots of a polynomial. Over the years, many modifications to Laguerre's method have been suggested in an attempt to improve its convergence rate and to avoid multiple convergence to a simple root. Now, we present a modification to Laguerre's method for the simultaneous convergence of all roots of a polynomial. Multiple numerical experiments verify both the accuracy and efficiency of this algorithm, and we provide comparisons to other root finding methods, such as PZEROS and AMVW. Immediate future research includes applying this method to the polynomial eigenvalue problem, where it is effective for both large degree problems and the Tridiagonal problem.

Introduction

Let $p(\lambda)$ be a polynomial of degree m. Denote by $(z_1, \ldots, z_m)$ current approximations to the roots, $r_1, \ldots, r_m$ of $p(\lambda)$. Then, for $j \in \{1, \ldots, m\}$ define

$$f_j(\lambda) = \frac{p(\lambda)}{\prod_{\substack{i=1 \\ i \neq j}}^{m} (\lambda - z_i)},$$

and note that $f_j(\lambda)$ has the same roots as $p(\lambda)$. Further define

$$G_j(\lambda) = \frac{f_j'(\lambda)}{f_j(\lambda)} = \frac{p'(\lambda)}{p(\lambda)} - \sum_{\substack{i=1 \\ i \neq j}}^{m} \frac{1}{\lambda - z_i},$$
$$H_j(\lambda) = -\left(\frac{f_j'(\lambda)}{f_j(\lambda)}\right)' = -\left(\frac{p'(\lambda)}{p(\lambda)}\right)' - \sum_{\substack{i=1 \\ i \neq j}}^{m} \frac{1}{(\lambda - z_i)^2}. \tag{1}$$

Then the jth root approximation is updated via the expression

$$\hat{z}_j = z_j - L_m(z_j), \tag{2}$$

where the modified Laguerre updated is defined by

$$L_m(z_j) = \frac{m}{G_j(z_j) \pm \sqrt{(m-1)(mH_j(z_j) - G_j^2(z_j))}}, \tag{3}$$

and the sign is chosen to maximize the magnitude of the denominator.

The removal of previously computed roots is a well known modification of Laguerre's method and has been employed in [4, 5]. Here, we create poles at $z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_m$ in order to direct the Laguerre iteration away from other root approximations. A similar idea was used by Aberth to modify Newton's method [1]. This makes for an algorithm that simultaneously approximates all roots of a polynomial.

Algorithm Outline

For $j = 1, \ldots, m$, the *jth* root approximation is updated via (2), provided that it is not already close enough to the root r_j. The iteration can be implemented in either a Jacobi style (can be parallelized) or in a Gauss-Seidel style (experiences slightly faster convergence). Our current algorithm (DSLM) uses a Gauss-Seidel iteration style and has the following attributes.

Order of Convergence. Locally, if the root is simple, convergence is quartic; otherwise it is linear.

Cost Complexity. The computation of both expressions in (1) requires the evaluation of $p(\lambda)$, $p'(\lambda)$, and $p''(\lambda)$, along with the evaluation of two sums; this can be done using Horner's rule in $O(m)$ time. The convergence of all roots requires $O(m)$ iterations, it follows that DSLM can compute all root approximations in $O(m^2)$ time.

Avoiding Overflow. If $|z_j| > 1$, then the computation of both expressions in (1) is slightly modified using the reversal polynomial of $p(\lambda)$ in order to avoid the potential overflow in Horner's rule.

Initial Estimates

Let $p(\lambda) = \sum_{i=0}^{m} a_i \lambda^i$, where $a_0 a_m \neq 0$. The *Newton polygon* associated with this polynomial is the upper convex hull of the discrete set $\{(i, \log|a_i| : \ i = 0, 1, \ldots, m\}$. Let $0 = k_0 < k_1 < \cdots < k_q = m$ denote the abscissas of the vertices of the Newton polygon, and define the radii

$$\mu_i = \left|\frac{a_{k_{i-1}}}{a_{k_i}}\right|^{\frac{1}{k_i - k_{i-1}}}, \ (i = 1, \ldots, q).$$

Then, $(k_i - k_{i-1})$ evenly distributed points are placed on circles centered at 0 with radius μ_i. These points constitute our initial estimates to the roots of $p(\lambda)$. Furthermore, these points are guaranteed to lie within the Pellet bounds for the polynomial $p(\lambda)$, and can be computed in $O(m \log m)$ time [3].

Stopping Criteria

It follows from [6][Lemma 3] that the backward error in the root approximation z_j is bounded above by

$$b(z_j) = \frac{|p(z_j)|}{\sum_{i=0}^{m} |a_i||z_j|^i}.$$

Criterion 1: If $b(z_j) < \epsilon$, where ϵ denotes double precision unit roundoff, then we say that the approximation z_j has converged.

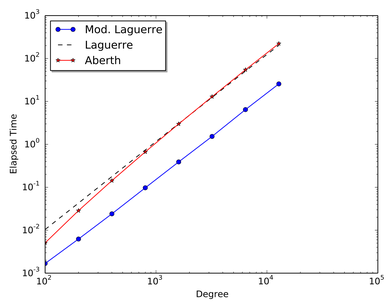
If Criterion 1 does not hold, then we compute the modified Laguerre update (3).

Criterion 2: If $|L_m(z_j)| < \epsilon|z_j|$, then no relatively significant contribution will be made by $L_m(z_j)$ and we say that the approximation z_j has converged.

If Criterion 2 does not hold, then we update z_j via (2).

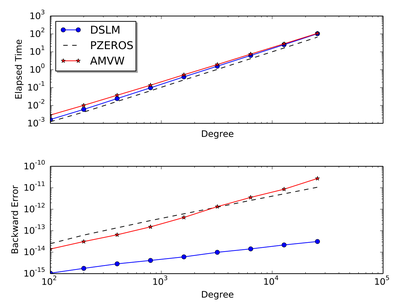
Comparison to Aberth and Laguerre

Elapsed time comparisons are made between our modified Laguerre method, the standard Laguerre method, and the Aberth method. The initial conditions and stopping criteria are the same for all methods. For each Degree, there are 25 different random polynomials created and the average elapsed time to compute the roots of each polynomial is recorded.



Comparison to PZEROS and AMVW

Random Polynomial Tests. Elapsed time and backward error comparisons are made between our method (DSLM), PZEROS [3], and AMVW [2]. For each degree, there are 25 different random polynomials created and the average elapsed time to compute the roots of each polynomial is recorded. In addition, the average of the maximum backward error for the roots of each polynomial is recorded.



Select Polynomial Tests. Relative forward error comparisons are made for the roots of the following polynomials.

Test No.	Description	Deg.	Roots
1	Wilkinson	10	$1, 2, \ldots, 10$
2	Wilkinson	15	$1, 2, \ldots, 15$
3	Wilkinson	20	$1, 2, \ldots, 20$
4	scaled and shifted Wilkinson	20	$-2.1, -1.9, \ldots, 1.7$
5	reverse Wilkinson	10	$1, 1/2, \ldots, 1/10$
6	reverse Wilkinson	15	$1, 1/2, \ldots, 1/15$
7	reverse Wilkinson	20	$1, 1/2, \ldots, 1/20$
8	prescribed roots of varying scale	20	$2^{-10}, 2^{-9}, \ldots, 2^{9}$
9	prescribed roots of varying scale - 3	20	$(2^{-10} - 3), (2^{-9} - 3), \ldots, (2^{9} - 3)$
10	Chebyshev polynomial	20	$\cos(\frac{2j-1}{40}\pi)$

	DSLM	POLZEROS	AMVW
Test 1	2.49E-11	1.05E-09	9.27E-11
Test 2	1.05E-07	1.57E-05	1.19E-05
Test 3	2.85E-03	3.49E-02	7.40E-02
Test 4	4.89E-13	5.52E-13	1.73E-11
Test 5	7.93E-12	9.65E-11	4.72E-07
Test 6	5.00E-08	2.41E-07	5.14E-02
Test 7	2.76E-04	6.26E-03	3.47E-01
Test 8	9.77E-04	9.77E-04	9.77E-04
Test 9	1.12E-03	1.30E-03	1.30E-03
Test 10	2.19E-12	1.46E-10	6.52E-11

Conclusion

Future research includes applying this modified Laguerre method to the polynomial eigenvalue problem, as was done in [4, 5]. In addition, we are interested in developing a parallelized version of DSLM using the Jacobi style iteration.

References

[1] O. Aberth, *Iteration methods for finding all zeros of a polynomial simultaneously*, Mathematics of Computation, 27 (1973), pp. 339–344.

[2] J. L. Aurentz, T. Mach, R. Vandebril, and D. S. Watkins, *Fast and backward stable computation of roots of polynomials*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 942–973.

[3] D. Bini, *Numerical computation of polynomial zeros by means of aberth's method*, Numerical Algorithms, 13 (1996), pp. 179–200.

[4] P. Lancaster, *Lambda-Matrices and Vibrating Systems*, vol. 94 of International Series of Monographs on Pure and Applied Mathematics, Pergamon, 1966.

[5] B. Parlett, *Laguerre's method applied to the matrix eigenvalue problem*, Mathematics of Computation, 18 (1964), pp. 464–485.

[6] F. Tisseur, *Backward error and condition of polynomial eigenvalue problem*, Linear Algebra and its Applications, 309 (2000), pp. 339–361.