Finding roots of real polynomial simultaneously by means of Bairstow's method

Article *in* BIT. Numerical mathematics · January 1998

DOI: 10.1007/BF01731985 · Source: CiteSeer

1 author:

Wai-Shing Luk
Fudan University
29 PUBLICATIONS **599** CITATIONS

SEE PROFILE

BIT 35 (1995), 001–003.

Finding roots of real polynomial simultaneously by means of Bairstow's method *

W. S. Luk[1]

[1] *Department of Computer Science and Engineering,*
The Chinese University of Hong Kong,
Shatin, N.T., Hong Kong.
email: luk036@cs.cuhk.hk

Abstract.

Aberth's method for finding the roots of a polynomial was shown to be robust. However, complex arithmetic is needed in this method even if the polynomial is real, because it starts with complex initial approximations. A novel method is proposed for real polynomials that does not require any complex arithmetic within iterations. It is based on the observation that Aberth's method is a systematic use of Newton's method. The analogous technique is then applied to Bairstow's procedure in the proposed method. As a result, the method needed half computations per iteration than Aberth's method. Numerical experiments showed that the new method exhibited a competitive overall performance for the test polynomials.

AMS subject classification: 65H05

Key words: Parallel algorithm, Aberth's method, Bairstow's method, polynomial zeros, implicit deflation.

1 Introduction.

Consider a problem of finding the roots of a real polynomial $P(x)$, deflation is a standard technique that was described in many numerical textbooks. An alternative way is to use the formula given by Weierstrass (see [10] and references therein):

$$x_i^{(n+1)} = x_i^{(n)} - \frac{P(x_i^{(n)})}{\prod_{j \neq i}^{N} (x_i^{(n)} - x_j^{(n)})} \qquad i = 1, \cdots, N, \tag{1.1}$$

where N is the degree of polynomial and $^{(n)}$ denotes the iteration index, for finding zeros simultaneously, Note that the convergence order of this

*Prepare for publication

method is quadratic. A related method for finding the quadratic factors was given by Dvorcuk [6]. In 1973, Aberth proposed another method that has cubic convergent rate [1]:

$$x_i^{(n+1)} = x_i^{(n)} - \frac{P(x_i^{(n)})}{P'(x_i^{(n)}) - \sum_{j\neq i}^{N} \frac{P(x_i^{(n)})}{(x_i^{(n)} - x_j^{(n)})}} \qquad i = 1, \cdots, N, \tag{1.2}$$

where $P'(x)$ denotes the derivative of $P(x)$. These methods have recently been investigated for parallel implementations [5, 4, 7]. In the previous discussion [3], it was pointed out that Aberth's method can be viewed as a modification of Maehly's procedure [11, pp. 259] by merely replacing the computed zeros with all other iterates. We call this generalized idea *Parallel Anticipatory Implicit Deflation* (PAID) approach.

As Aberth mentioned, using Aberth's method with asymmetric iterates can overcome the symmetric problem [1]. However, complex arithmetic will not be avoided in this configuration even when the polynomial is real. The possibility of using this method with symmetric iterates will not be considered in this paper. Instead, we attempt to overcome this deficiency using PAID idea and Bairstow's method. The idea is similar to the one in [13], but the resulting method is simplier here, which makes it more competitive with Aberth's method.

Recall that Bairstow's method avoids complex arithmetic by seeking the quadratic factor $(x^2 - rx - q)$ of $P(x)$ [11, pp. 301–303]. Let (A, B) be the coefficients of the linear remainder of $P(x)$ such that:

$$P(x) = (x^2 - rx - q)P_1(x) + Ax + B, \tag{1.3}$$

and (A_1, B_1) be the coefficients of the linear remainder of $P_1(x)$ such that:

$$P_1(x) = (x^2 - rx - q)P_2(x) + A_1x + B_1. \tag{1.4}$$

Bairstow's method can be written as:

$$\begin{bmatrix} r^{(n+1)} \\ q^{(n+1)} \end{bmatrix} = \begin{bmatrix} r^{(n)} \\ q^{(n)} \end{bmatrix} - \begin{bmatrix} A_1 r^{(n)} + B_1 & A_1 \\ A_1 q^{(n)} & B_1 \end{bmatrix}^{-1} \begin{bmatrix} A \\ B \end{bmatrix}. \tag{1.5}$$

Horner-type scheme is used to evaluate (A, B) and (A_1, B_1). By following PAID approach, we construct a parallel method as follows. First, a method of suppression of computed quadratic factors is employed. The details will be given in §2. Next, by replacing the computed quadratic factors with the trial factors to the suppression method, we obtain a simultaneous version of Bairstow's method that is described in §3. We will

also discuss a simple method for choosing the initial guesses in §4. Numerical results will be presented in §5. For simplicity, we will omit the superscript $^{(n)}$ in the following sections if it is understood.

2 Suppression of computed quadratic factors.

The first step of developing the novel algorithm is to find out a method of suppression, which was described in [8]. Assume that $(x^2 - \tilde{r}x - \tilde{q})$ has been found to be a factor of $P(x)$. Let $\tilde{P}(x) = P(x)/(x^2 - \tilde{r}x - \tilde{q})$ be the deflated polynomial. The goal of suppression is to perform the Bairstow process without explicitly constructing $\tilde{P}(x)$. Let $(\tilde{A}, \tilde{B})$ be the coefficients of the linear remainder of $\tilde{P}(x)$ such that:

$$\tilde{P}(x) = \tilde{P}_1(x)(x^2 - rx - q) + \tilde{A}x + \tilde{B} \tag{2.1}$$

and $(\tilde{A}_1, \tilde{B}_1)$ be the coefficients of the linear remainder of $\tilde{P}_1(x)$ such that:

$$\tilde{P}_1(x) = \tilde{P}_2(x)(x^2 - rx - q) + \tilde{A}_1 x + \tilde{B}_1. \tag{2.2}$$

The relation between A, B, A_1, B_1 and $\tilde{A}, \tilde{B}, \tilde{A}_1, \tilde{B}_1$ can be expressed in the form [8]:

$$\begin{cases} p = r - \tilde{r}, & l = q - \tilde{q} \\ f = r \cdot p + l, & e = f \cdot l - q \cdot p^2 \\ a = A \cdot l - B \cdot p, & b = B \cdot f - A \cdot q \cdot p \\ c = A_1 \cdot e - a, & d = B_1 \cdot e - b - a \cdot p \\ \tilde{A} = a \cdot e, & \tilde{B} = b \cdot e \\ \tilde{A}_1 = c \cdot l - d \cdot p, & \tilde{B}_1 = d \cdot f - c \cdot q \cdot p \end{cases} \tag{2.3}$$

where a, b, c, d, e, f, l and p are intermediate variables. A second quadratic factor can be suppressed by repeating the process starting with $\tilde{A}, \tilde{B}, \tilde{A}_1, \tilde{B}_1$, and so on.

3 The modified Bairstow method.

A simultaneous version of Bairstow's method can now be obtained by replacing $(\tilde{r}, \tilde{q})$ in (2.3) with the trial factors in case of even–degree polynomials, i.e., $N = 2M$. Starting with M trial factors $(x^2 - r_i^{(0)}x - q_i^{(0)})$, $i = 1, \cdots M$, the Bairstow iteration is applied to each trial factor in parallel, treating all the other factors as the computed ones and performing the suppression process according to (2.3). The modified Bairstow method is summarized as shown in Fig 3.1.

ALGORITHM: The modified Bairstow method

Select a set of initial guesses (r_i, q_i), $i = 1, \cdots, M$.
Repeat
  Do $i = 1$ to M in parallel,
    Evaluate A, B, A_1, B_1 using (r_i, q_i).
    Do $j = 1$ to M, $j \neq i$,

$$p \leftarrow r_i - r_j, \qquad l \leftarrow q_i - q_j$$
$$f \leftarrow r_i \cdot p + l, \qquad e \leftarrow f \cdot l - q_i \cdot p^2$$
$$a \leftarrow A \cdot l - B \cdot p, \qquad b \leftarrow B \cdot f - A \cdot q_i \cdot p$$
$$c \leftarrow A_1 \cdot e - a, \qquad d \leftarrow B_1 \cdot e - b - a \cdot p$$
$$A \leftarrow a \cdot e, \qquad B \leftarrow b \cdot e$$
$$A_1 \leftarrow c \cdot l - d \cdot p, \qquad B_1 \leftarrow d \cdot f - c \cdot q_i \cdot p$$

    End Do
    Update (r_i, q_i) by Bairstow's method.
  End Do
Until converge.

Figure 3.1: Outline of the modified Bairstow method

In case of odd–degree polynomials, we need a special treatment as follows. An extra root is added to the polynomial such that the resulting polynomial is even–degree before applying to the formulas. In our scheme, we choose the root at origin as the extra root for convenience. Therefore, if there are any roots at origin in the original polynomial, they should be removed at the beginning and be remembered as part of the solution. The overall algorithm is summarized as follows.

1. Remove any roots at origin from the polynomial and remember them as part of the solution.

2. If the resulting polynomial is odd–degree, insert a root at origin to make the degree even.

3. Apply the iteration according to the algorithm as shown in Fig. 3.1 to find the roots.

4. If the original polynomial is odd-degree, remove the root at origin that was inserted at step 2.

5. Add the roots at origin that was stored at step 1.

Table 3.1: Computations required per iteration (up to the leading terms) of Aberth's method and the modified Bairstow method. N denotes the degree of a polynomial and $N = 2M$.

	Aberth's method		The Proposed method	
	Mult/Div	Add/Sub	Mult/Div	Add/Sub
Horner	$8N^2$	$7N^2$	$4NM$	$4NM$
Suppression	$4N^2$	$4N^2$	$19M^2$	$11M^2$
Total	$12N^2$	$11N^2$	$27M^2$	$19M^2$

At first sight, it seems that the method is worse than Aberth's method because of the expensive cost of quadratic factors suppression. However, as we mentioned before, since complex arithmetic is avoided, the method is in fact more economical. Table 3.1 shows the computations required per iteration of two methods. Only the leading terms are counted. The first row of the table indicates the cost of Horner-type evaluations. It includes the evaluations of $P(x_i)$ and $P'(x_i)$ in Aberth's method and the evaluations of (A, B) and (A_1, B_1) in the proposed method. The second row indicates the cost of *zeros/quadratic factors* suppression. It includes the evaluations of $1/(x_i - x_j)$ in Aberth's method and the evaluations of equation (2.3) in the proposed method. If addition, subtraction, multiplication or division is counted as one *flop* (floating-point operation), the total *flops* per iteration of Aberth's method are $23N^2 + O(N)$ and those of the proposed method are $46M^2 + O(N)$. Hence roughly a factor of two improvement will be expected.

4 Choosing the initial guesses.

We follow Aberth's suggestion that the initial guesses should be evenly distributed on a circle with the center C which equals to the centroid of zeros [1]:

$$z_j^{(0)} = C + R \cdot \exp(2\pi ij/N + i\phi) \qquad j = 1, \cdots, N,$$

where $i = \sqrt{-1}$. C can be determined by the formula $-a_{N-1}/(N \cdot a_N)$, where a_N and a_{N-1} represent the coefficients of x^N and x^{N-1} respectively. The angle ϕ is used to break the symmetry with respect to the real axis, which is taken as $\pi/2N$ in Aberth's discussion. For the parallel Bairstow method, we just put it as zero. R is taken as the effective radius R_e by Chen [2]:

$$R = R_e = (-P(C))^{1/N}$$

for Aberth's method. Note that R_e is a complex number. For the proposed method, since R must be a real number and therefore we use $R = |P(C)|^{1/N}$ instead. As a result, the initial $\{r_j^{(0)}, q_j^{(0)}\}$ are given by:

$$\begin{cases} r_j^{(0)} = 2\, real(z_j^{(0)}) \\ q_j^{(0)} = |z_j^{(0)}|^2 \end{cases} \qquad j = 1, \cdots, M.$$

5 Numerical experiments.

We have written experimental programs in MATLAB to implement the modified Bairstow's method described in Section 3 and Aberth's method. The number of iterations (*iter.*) and the floating-point operation count (*flops*) were used to measure the performance. Since the stopping criterions of these two methods were different, we listed the maximum of the actual residual error ($\xi_{\max}$), which is defined as $\max_i |P(x_i)|$, for comparison. We follow [7] that the test polynomials were taken from Table 1 of [9] (see also [12]).

The programs were run on a DECstation 5000/133, from Digital Equipment Corporation and the results are listed in Table 5.1. We observed that the convergent rates of two methods were quite similar except for some polynomials. At the following, we will discuss those problems. In Problem 3, 5, 6, 7 and 9, slower convergent rates were observed in Aberth's method. It was because the corresponding polynomials contained either multiple roots or only real roots or both. However, the modified Bairstow method may not in all cases exhibit fast convergence. For example, in Problem 10, a slow convergent rate was also found. In Problem 31, the modified Bairstow method required 50 iterations to converge. We observed that in the first 44 iterations, one pair of iterates was just wandering but suddenly approached the roots quickly at the last several iterations. An appropriate explanation is still unknown however.

6 Conclusions.

In this paper, the general idea of the proposed method has been presented. The advantage of this method is that complex arithmetic can be avoided and inherently parallelizable. Numerical results have shown the robustness and the efficiency of this method.

Acknowledgement.

The author would like to thank Tien-Chi Chen and Kei-Shiu Ho for many helpful suggestions and discussions.

REFERENCES

1. O. Aberth, *Iteration methods for finding all zeros of a polynomial simultaneously*, Math. Comp., 27 (1973), pp. 339–344.
2. T.-C. Chen, *SCARFS, an efficient polynomial zero-finder system*, in Proc. Int'l Conf. on APL, Toronto, Canada, Aug. 1993, pp. 47–54.
3. T.-C. Chen and W.-S. Luk, *Aberth's method for the parallel iterative finding of polynomial zeros*, in *APL*94 Conference Proceedings, Antwerp, Belgium, Sept. 1994, pp. 40–49.
4. M. Cosnard and P. Fraigniaud, *Finding the roots of a polynomial on a MIMD multicomputer*, Parallel Computing, 15 (1990), pp. 75–85.
5. ——, *Analysis of asynchronous polynomial root finding methods on a distributed memory multicomputer*, IEEE trans. parallel and distrib. syst., 5 (1994), pp. 639–648.
6. J. Dvorcuk, *Factorization of a polynomial into quadratic factors by Newton method*, Aplikace Matematiky, 14 (1969), pp. 54–80.
7. T. L. Freeman, *Calculating polynomial zeros on a local memory parallel computer*, Parallel Computing, 12 (1989), pp. 351–358.
8. D. C. Handscomb, *Computation of the latent roots of a Hessenberg matrix by Bairstow's method*, Computer Journal, 5 (1962), pp. 139–141.
9. P. Henrici and B. O. Watkins, *Finding zeros of a polynomial by the Q–D algorithm*, Commun. ACM, 8 (1965), pp. 570–574.
10. N. Kjurkchiev and K. Mahdi, *Some remarks on Dvorcuk's root-finding method*, BIT, 34 (1994), pp. 318–322.
11. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer–Verlag, 2nd ed., 1993.
12. R. F. Thomas, *Corrections to numerical data on Q–D algorithm*, Commun. ACM, 9 (1966), pp. 322–323.
13. S.-M. Zheng, *Linear interpolation and parallel iteration for splitting factors of polynomials*, J. of Comput. Math., 4 (1986), pp. 146–153.

Table 5.1: Comparison between Aberth's method and the modified Bairstow method

Problem	Degree.	Aberth's method			The proposed method		
		iter.	*flops*	$\xi_{\max}$	*iter.*	*flops*	$\xi_{\max}$
1	3	4	1339	8.88e-16	5	1170	5.02e-15
2	3	4	1335	1.95e-13	5	1143	2.45e-15
3	3	10	2902	2.25e-16	4	896	2.22e-16
4	3	6	1867	2.22e-16	16	3473	1.87e-14
5	3	10	2919	4.44e-16	5	1065	4.44e-16
6	3	10	2902	4.09e-11	4	893	7.11e-15
7	4	32	15019	3.40e-14	4	941	1.02e-14
8	4	5	2709	6.79e-14	5	1161	1.29e-13
9	4	12	5880	6.44e-11	5	1066	0
10	4	12	5880	4.18e-11	12	2604	4.33e-13
11	4	11	5424	2.51e-11	17	3648	6.22e-15
12	4	12	5878	2.25e-11	16	3476	1.07e-14
13	4	9	4512	1.14e-13	8	1698	1.14e-13
14	4	15	7248	1.65e-11	14	2966	5.26e-13
15	4	5	2709	1.07e-11	5	1169	6.68e-16
16	4	7	3598	5.73e-14	5	1108	5.73e-14
17	4	4	2230	5.62e-12	8	1696	4.26e-14
18	5	4	3386	3.91e-14	5	2386	5.47e-14
19	6	4	4760	5.55e-16	7	3154	4.97e-16
20	6	6	6793	3.49e-11	7	3279	3.68e-11
21	6	8	8764	6.23e-13	8	3678	5.30e-12
22	6	7	7768	5.66e-12	7	3225	7.11e-12
23	6	9	9760	2.34e-13	8	3704	6.54e-12
24	6	6	6772	7.56e-12	8	3679	2.57e-12
25	7	7	10446	3.33e-13	9	7188	9.77e-14
26	7	14	19875	414	17	13663	414
27	10	9	26256	8.34e-3	9	11180	8.34e-3
28	12	8	33661	2.12e+20	11	19991	3.44e+15
29	13	8	39354	3.14e+22	23	55932	1.33e+23
30	15	10	64126	2	14	43380	2.93
31	18	9	83269	6.05e-07	50	191307	1.24e-06
32	19	9	92643	2.27e-06	7	33610	3.77e-06