Finding the roots of a real polynomial simultaneously using Bairstow's method

Wai-Shing Luk

# Abstract

Aberth's method of finding the roots of polynomials is shown to be robust. However, in this method, complex arithmetic is required even if the polynomial is real, since it starts with a complex initial approximation. For real polynomials, we propose a new method that does not require any complex arithmetic in the iterative process. It is based on the observation that Aberth's method is a system use of Newton’s methods. Then, in the proposed method, a similar technique is applied to Bairstow's procedure. As a result, the method requires half as much computation per iteration as Aberth's method. Numerical experiments show that the new method exhibits competitive overall performance for the polynomials tested.

## Introduction

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

where is the degree of the polynomial and denotes the iteration index, used to find the zeros simultaneously. Note that the order of convergence of this method is quadratic. A related method for finding quadratic factors was given by Dvorciuk [6]. In 1973, Aberth proposed another method with a cubic rate of convergence [1].

which denotes the dervative of . These methods have recently been studied for parallel implementations [5, 4, 7]. In a previous discussion [3], it was pointed out that Aberth's method can be seen as a modification of Maehly's proceduce [11, p. 259], simply replacing the computed zeros with all other iterates. We call this generalized idea the *Parallel Anticipatory Implicit Deflation* (PAID) method.

As mentioned by Aberth, the symmetry problem can be overcome by using asymmetric iterates for Aberth's method [1]. However, in this configuration, complex arithmetics cannot be avoided even if the polynomial is real. In this paper, we will not consider the possibility of symmetric iterates using this approach. Instead, we try to use the PAID idea and Bairstow's method to overcome this drawback. The idea is similar to the one in the literature [13], but the resulting method is simpler, which makes it more competitive with Aberth's method.

Recall that Bairstow's method avoids complex arithmetic by seeking quadratic factors of [11, pp. 301-303]. Let be the coefficients of the linear remainder of such that:

and the coefficients of the linear remainder of such that:

Bairstow's method can be written as:

Horner-type scheme is used to evaluate and . By following the PAID approach, we construct a parallel method as follows. First, we use a method that suppresses the computation of quadratic factors. Details will be given in Section 2. Next, by replacing the computed quadratic factors with the trial factors of the suppression method, we obtain a simultaneous version of the Bairstow method, which will be described in Section 3. We will also discuss a simple method for selecting initial guesses in Section 4. Numerical results will be presented in Section 5. For the sake of simplicity, we will omit the superscript We will omit superscripts in the following sections if they are understood.

## 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 has been found to be a factor of . Let be that deflated polynomial. The goal of suppression is to perform the Bairstow process without explicitly consructing . Let be the coefficients of the linear remainder of and be the coefficients of the linear remainder of . The relation between and can be expressed in the form [8]:

where , , , , , , , and are intermediate variables. A second quadratic factor can be suppressed by repeating the process starting with and so on.

## Modified Bairstow method

A simultaneous version of Bairstow’s method can now be obtained by replacing in (2.3) with the trial factors in case of even-degree polynomials, i.e., . Starting with trial factors , , the Bairstow iteration is applied to each trial factor in parallel, treating all other factors as computed and performing the suppression process according to (2.3). An overview of the modified Bairstow method is shown in Figure 3.1.

In the case of polynomials of odd degree, we need to make the following special treatment. Before applying to the formula, an extra root is added to the polynomial so that the resulting polynomial is of even degree. In our scheme, for convenience, we choose the root of the origin as the extra root. Thus, if there are any roots of the origin in the original polynomial, they should be removed at the beginning and remembered as part of the solution. The entire algorithm is summarized below.

1. Remove any roots at the origin from the polynomial and remember them as part of the solution.
2. If the resulting polynomial is of odd degree, insert a root at the origin to make it of even degree.:
3. Iterate according to the algorithm shown in Figure 3.1 to find the root.
4. If the original polynomial is of odd degree, delete the root at the origin inserted in step 2.
5. Add the roots at the origin stored in step 1.

At first glance, the method appears to be inferior to Aberth's method because of the high cost of suppressing the quadratic factors. However, as we mentioned before, the method is actually more economical due to the avoidance of complex arithmetic. Table 3.1 shows the amount of computation required per iteration for both methods. Only the leading terms are counted. The first row of the table shows the cost of the Horner type of evaluation. It includes the evaluation of and of the evaluation, and in Aberth’s method for and of the evaluation. The second row indicates the cost of suppressing the zeros/quadratic factors.

It includes the evaluations of in Aberth’s method and the evaluations of equation (2.3) in the proposed method. Inf addition, subtraction, multiplication or division is counted as one *flop* (floationg-point operation), the total flops per iteration of Aberth’s method are and those of the proposed method are . Hence roughly a factor of two improvement will be expected.

## Choosing the initial guesses

We follow Aberth’s suggestion that the initial guesses should be evenly distributed on a circle with the center which euqals to the cenetroid of zeros [1]:

where . can be determined by the formula , where and represent the coefficients of and respectively. The angle is used to break the symmetry with respect to the real aixs, which is taken as in Aberth’s discussion. For the parallel Bairstow method, we just put it as zero. is taken as the effective radius by Chen [2]:

for Aberth's method. Note that is a complex number. For the proposed method, since must be a real number and therefore we use instead. As a result, the initial are given by:

## 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 (), which is defined as , for comparison. We follow [7] that the test polynomials were taken form Table 1 of [9] (see also [12]).

The programs were run on a DECstation 5000/133, form Digital Equipment Corporation and the results are listed in Table 5.1. We observed that the convergent rates of the 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 howerver.

## 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.