Electronic Transactions on Numerical Analysis.
Volume 55, pp. 401–423, 2022.
Copyright © 2022, Kent State University.
ISSN 1068–9613.
DOI: 10.1553/etna_vol55s401

ETNA
Kent State University and
Johann Radon Institute (RICAM)

ON A COMPENSATED EHRLICH-ABERTH METHOD FOR THE ACCURATE COMPUTATION OF ALL POLYNOMIAL ROOTS[*]

THOMAS R. CAMERON[†] AND STEF GRAILLAT[‡]

Abstract. In this article, we use the complex compensated Horner method to derive a compensated Ehrlich-Aberth method for the accurate computation of all roots, real or complex, of a polynomial. In particular, under suitable conditions, we prove that the limiting accuracy for the compensated Ehrlich-Aberth iterations is as accurate as if computed in twice the working precision and then rounded to the working precision. Moreover, we derive a running error bound for the complex compensated Horner method and use it to form robust stopping criteria for the compensated Ehrlich-Aberth iterations. Finally, extensive numerical experiments illustrate that the backward and forward errors of the root approximations computed via the compensated Ehrlich-Aberth method are similar to those obtained with a quadruple precision implementation of the Ehrlich-Aberth method with a significant speed-up in terms of computation time.

Key words. polynomial evaluation, error-free transformations, polynomial roots, backward error, forward error, rounding error analysis

AMS subject classifications. 65H04, 65Y20, 65-04

1. Introduction.

The use of error-free transformations to produce compensated arithmetic routines has a long and interesting history, which includes the works of Dekker, Gill, Goldberg, Kahan, Knuth, and Møller [6, 12, 13, 21, 22, 23]. These works were the first to extend the working precision of a computation without the use of a hardware or software implementation of a high precision format [9, 10]. More recently, Rump, Ogita, and Oishi have developed algorithms for the summation and the dot product computed in k-fold working precision [25], $k \geq 2$, as well as the computation of a summation that is faithfully rounded [27, 28].

In addition, error-free transformations have been used to develop the compensated Horner method for the evaluation of a polynomial and its derivatives at a real or complex number [15, 16, 17, 19, 20]. The real compensated Horner method has been used to improve the accuracy of eigenvalue approximations of a symmetric tridiagonal matrix and real root approximations of a real polynomial [11, 14]. In [14], Graillat shows that if the real compensated Horner method is used to evaluate a polynomial, then the Newton iterations converge to an approximate root as accurately as if computed in twice the working precision and then rounded to the working precision.

The Ehrlich-Aberth (Börsch-Supan) method [1, 4, 8] combines Newton's method with an implicit deflation strategy, which allows for the computation of all roots, real or complex, of a polynomial. In this article, we use the complex compensated Horner method to develop a compensated Ehrlich-Aberth method that can compute all roots of a polynomial as accurately as the Newton iterations from [14]. The outline of this article is as follows. In Section 2, we recall the basic properties of real and complex floating-point arithmetic. Then, in Section 3, we describe the Horner method and the compensated Horner method for polynomial evaluation in complex floating-point arithmetic. Moreover, we derive a running error bound for the compensated Horner method, which takes into account the rounding errors that occur during

[*] Received September 12, 2021. Accepted February 10, 2022. Published online on March 21, 2022. Recommended by Dario Bini. This work was partly supported by the NuSCAP (ANR-20-CE48-0014) project of the French National Agency for Research (ANR).

[†] Department of Mathematics, Penn State Behrend, Erie, PA (trc5475@psu.edu).
https://behrend.psu.edu/person/thomas-cameron-phd

[‡] Sorbonne UniversitÃľ, CNRS, LIP6, F-75005 Paris, France
(stef.graillat@sorbonne-universite.fr). http://www-pequan.lip6.fr/~graillat

the computation. Section 4 is devoted to the presentation of the Ehrlich-Aberth method and the compensated Ehrlich-Aberth method. In particular, the running error bound for the compensated Horner method is used to form robust stopping criteria for the sequence of root approximations. Moreover, under suitable conditions, we prove that these root approximations have a similar limiting accuracy as if computed in twice the working precision and then rounded to the working precision. Finally, in Section 5, extensive numerical experiments illustrate that the backward and forward errors of the root approximations computed via the compensated Ehrlich-Aberth method are similar to those obtained with a quadruple precision implementation of the Ehrlich-Aberth method but with a significant speed-up in terms of computation time.

2. Floating-point arithmetic.

Throughout this article, we assume that the computer arithmetic satisfies the IEEE 754 standard [2] and that neither underflow nor overflow occurs. We denote by $\mathcal{F}$ the set of floating-point numbers and by μ the unit roundoff. Note that for single precision, $\mu = 2^{-24}$ and for double precision, $\mu = 2^{-53}$, where the exponent corresponds to the precision of this floating-point format. Finally, we use the standard notation $\mathrm{fl}(\cdot)$ to denote floating-point operations in working precision.

2.1. Real floating-point arithmetic.

For operations $\circ \in \{+, -, \cdot\}$, the IEEE 754 standard requires the result of $\mathrm{fl}(a \circ b)$ to be *correctly rounded*, i.e., as accurate as if computed exactly and then rounded to the current precision [13]. In this article, we assume that all the computations are performed with rounding to nearest (using round to even). As a result, for $a, b \in \mathcal{F}$, the floating-point operations satisfy

$$\mathrm{fl}(a \circ b) = (a \circ b)(1 + \epsilon),$$

where $|\epsilon| \leq \mu$. This further implies that

$$|\mathrm{fl}(a \circ b) - a \circ b| \leq \mu |a \circ b| \quad \text{and} \quad |a \circ b - \mathrm{fl}(a \circ b)| \leq \mu |\mathrm{fl}(a \circ b)|.$$

Throughout this article, we make use of the quantities γ_n, which are defined in the usual way [18]:

$$\gamma_n = \frac{n\mu}{1 - n\mu},$$

where $n \in \mathbb{N}$ is assumed to satisfy $n\mu < 1$. Moreover, we make use of the round to nearest mode to perform error-free transformations for a floating-point operation. In particular, for each $x = \mathrm{fl}(a \circ b)$, there exists a $y \in \mathcal{F}$ such that $x + y = a \circ b$. The pair (x, y) is called the *error-free transformation* of (a, b) for the operation $\circ$. For instance, Algorithm 1 is attributed to Knuth [22] and returns the error-free transformation of (a, b) for addition. Note that Algorithm 1 requires 6 flops to be executed.

Algorithm 1 Error-free transformation of $(a, b) \in \mathcal{F}^2$ for addition.

```
function $[x, y] = \mathrm{TwoSum}(a, b)$ :
    $x = \mathrm{fl}(a + b)$
    $z = \mathrm{fl}(x - a)$
    $y = \mathrm{fl}((a - (x - z)) + (b - z))$
```

In addition, we make use of the fused multiply-add operation, denoted $\mathrm{FMA}(a, b, c)$, which results in the floating-point number nearest to $a \cdot b + c \in \mathbb{R}$. Note that the fused multiply-add operation was added to the IEEE 754 standard in 2008 and is supported by many modern processors [24]. Moreover, the error-free transformation of (a, b) for multiplication can be performed using the fused multiply-add operation as is done in [25]; see Algorithm 2.

Algorithm 2 Error-free transformation of $(a, b) \in \mathcal{F}^2$ for multiplication.

```
function $[x, y] = \mathrm{TwoProduct}(a, b)$ :
  $x = \mathrm{fl}(a \cdot b)$
  $y = \mathrm{FMA}(a, b, -x)$
```

For processors that do not support the fused multiply-add operation, the error-free transformation of (a, b) for multiplication can be computed with the splitting operation introduced by Dekker [6] but requires 17 flops rather than the 2 flops needed by Algorithm 2; see [25, Algorithm 3.3]. We conclude this section with a theorem from [25] that summarizes the properties of Algorithm 1 and Algorithm 2.

THEOREM 2.1. *Let $a, b \in \mathcal{F}$. Then, for $[x, y] = \mathrm{TwoSum}(a, b)$, we have*

$$a + b = x + y, \qquad x = \mathrm{fl}(a + b), \qquad |y| \leq \mu|x|, \qquad |y| \leq \mu|a + b|,$$

and, for $[x, y] = \mathrm{TwoProduct}(a, b)$, we have

$$a \cdot b = x + y, \qquad x = \mathrm{fl}(a \cdot b), \qquad |y| \leq \mu|x|, \qquad |y| \leq \mu|a \cdot b|.$$

2.2. Complex floating-point arithmetic. We define $\mathcal{C} = \mathcal{F} + i\mathcal{F}$ to be the set of complex floating-point numbers, where $i = \sqrt{-1}$ is the imaginary unit. Also, we use the operators $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ to denote the real and imaginary part of a complex number, respectively. As in the real case, we denote by $\mathrm{fl}(\cdot)$ the operations that are done in floating-point working precision. The following holds for all $a, b \in \mathcal{C}$ and $\circ \in \{+, -\}$:

$$\mathrm{fl}(a \circ b) = (a \circ b)(1 + \epsilon),$$

where $|\epsilon| \leq \mu$. In addition, we have

$$\mathrm{fl}(a \cdot b) = (a \cdot b)(1 + \epsilon),$$

where $|\epsilon| \leq \sqrt{2}\,\gamma_2$. This further implies that

$$|\mathrm{fl}(a \circ b) - a \circ b| \leq \mu|a \circ b| \quad \text{and} \quad |a \circ b - \mathrm{fl}(a \circ b)| \leq \mu|\mathrm{fl}(a \circ b)|,$$

for $\circ \in \{+, -\}$, and

$$|a \cdot b - \mathrm{fl}(a \cdot b)| \leq \sqrt{2}\,\gamma_2|a \cdot b|.$$

Finally, throughout this article, we make use of the quantity

$$\widetilde{\gamma}_n = \frac{n\sqrt{2}\,\gamma_2}{1 - n\sqrt{2}\,\gamma_2}.$$

As in the real case, the error-free transformation of the pair of complex floating-point numbers (a, b) for the operation $\circ$ is a pair (x, y) such that $x = \mathrm{fl}(a \circ b)$ and $x + y = a \circ b$. The error-free transformation of $(a, b) \in \mathcal{C}^2$ for complex addition is a straightforward extension

Algorithm 3 Error-free transformation of $(a, b) \in \mathcal{C}^2$ for addition.

```
function $[x, y] = \mathrm{TwoSumCmplx}(a, b)$ :
    $[\mathrm{Re}(x), \mathrm{Re}(y)] = \mathrm{TwoSum}(\mathrm{Re}(a), \mathrm{Re}(b))$
    $[\mathrm{Im}(x), \mathrm{Im}(y)] = \mathrm{TwoSum}(\mathrm{Im}(a), \mathrm{Im}(b))$
```

of Algorithm 1 and is shown in Algorithm 3. Note that Algorithm 3 requires 12 flops to be executed.

The error-free transformation of $(a, b) \in \mathcal{C}^2$ for complex multiplication requires multiple products of the real and imaginary parts of a and b as illustrated in Algorithm 4. Note that Algorithm 4 requires 20 flops to be executed. In contrast, if the splitting operation from [6] was used in Algorithm 2, then Algorithm 4 would require 64 flops to be executed; see [17, Algorithm 3.3].

Algorithm 4 Error-free transformation of $(a, b) \in \mathcal{C}^2$ for multiplication.

```
function $[w, x, y, z] = \mathrm{TwoProductCmplx}(a, b)$ :
    $[g_1, h_1] = \mathrm{TwoProduct}(\mathrm{Re}(a), \mathrm{Re}(b))$; $[g_2, h_2] = \mathrm{TwoProduct}(\mathrm{Im}(a), \mathrm{Im}(b))$
    $[g_3, h_3] = \mathrm{TwoProduct}(\mathrm{Re}(a), \mathrm{Im}(b))$; $[g_4, h_4] = \mathrm{TwoProduct}(\mathrm{Re}(a), \mathrm{Im}(b))$
    $[g_5, h_5] = \mathrm{TwoSum}(g_1, -g_2)$; $[g_6, h_6] = \mathrm{TwoSum}(g_3, g_4)$
    $w = g_5 + ig_6$; $x = h_1 + ih_3$; $y = -h_2 + ih_4$; $z = h_5 + ih_6$
```

We conclude this section with a theorem from [17] that summarizes the properties of Algorithm 3 and Algorithm 4.

THEOREM 2.2. *Let $a, b \in \mathcal{C}$. Then, for $[x, y] = \mathrm{TwoSumCmplx}(a, b)$, we have*

$$a + b = x + y, \qquad x = \mathrm{fl}(a + b), \qquad |y| \leq \mu|x|, \qquad |y| \leq \mu|a + b|,$$

and, for $[w, x, y, z] = \mathrm{TwoProductCmplx}(a, b)$, we have

$$a \cdot b = w + x + y + z, \qquad w = \mathrm{fl}(a \cdot b), \qquad |x + y + z| \leq \sqrt{2}\,\gamma_2|a \cdot b|.$$

3. The Horner method.

Consider the polynomial of degree m in the variable z defined by

$$p(z) = a_m z^m + \cdots + a_1 z + a_0, \tag{3.1}$$

where $a_0, a_1, \ldots, a_m \in \mathcal{C}$, and $a_m \neq 0$. Given $z \in \mathcal{C}$, we can compute the polynomial evaluation $p(z)$ using the Horner method as shown in Algorithm 5.

Algorithm 5 Horner method.

```
function $[h_0] = \mathrm{Horner}(p, z)$ :
    $h_m = a_m$
    for $k = m - 1$ to $k = 0$ do
        $h_k = \mathrm{fl}(z \cdot h_{k+1} + a_k)$
    end for
```

The result $h_0 = \mathrm{Horner}(p, z)$ satisfies the following forward error bound [17]:

$$|p(z) - h_0| \leq \widetilde{\gamma}_{2m}\,\widetilde{p}(|z|),$$

where $\widetilde{p}(z) = \sum_{k=0}^{m} |a_k| z^k$. Finally, if $p(z) \neq 0$, then we have the relative forward error bound

$$\frac{|p(z) - h_0|}{|p(z)|} \leq \widetilde{\gamma}_{2m} \operatorname{cond}(p, z),$$

where the condition number of the polynomial evaluation of p at z is defined by

$$\operatorname{cond}(p, z) = \widetilde{p}(|z|)/|p(z)|. \tag{3.2}$$

3.1. The compensated Horner method. To improve the standard error bound for the Horner method, we follow the development in [17] to produce a compensated Horner method in complex floating-point arithmetic. To this end, we record the error at each iteration from both the floating-point product and the sum operations. Specifically, we produce four error polynomials: $p_\pi, p_\mu, p_\nu, p_\sigma$, which are collected monomial-by-monomial as outlined in Algorithm 6. Note that each iteration of Algorithm 6 requires 12 flops for Algorithm 3 and 20 flops for Algorithm 4 with a total of $32m$ flops.

Algorithm 6 Error-free transformation of the Horner method.

```
  function $[h_0, p_\pi, p_\mu, p_\nu, p_\sigma] = \mathrm{EFTHorner}(p, z)$ :
    $h_m = a_m$
    **for** $k = m - 1$ **to** $k = 0$ **do**
      $[\hat{h}_k, \pi_k, \mu_k, \nu_k] = \mathrm{TwoProductCmplx}(h_{k+1}, z)$
      $[h_k, \sigma_k] = \mathrm{TwoSumCmplx}\left(\hat{h}_k, a_k\right)$
      Set $\pi_k, \mu_k, \nu_k, \sigma_k$, respectively, as the coefficient of degree $k$ in $p_\pi, p_\mu, p_\nu, p_\sigma$.
    **end for**
```

It is immediately clear that $h_0 = \mathrm{Horner}(p, z)$. Furthermore, by induction, it is easy to show that

$$p(z) = h_0 + (p_\pi + p_\mu + p_\nu + p_\sigma)(z),$$

and it follows from [17, Proposition 5.3] that

$$((p_\pi \widetilde{+ p_\mu +} p_\nu) + \widetilde{p}_\sigma)(|z|) \leq \widetilde{\gamma}_{2m}\, \widetilde{p}(|z|).$$

To obtain a true error-free transformation, we would need to recursively perform Algorithm 6 on the four error polynomials of degree $(m - 1)$ until the resulting error polynomials were constant. This recursive process would result in many error terms: four polynomials of degree $(m - 1)$, 16 polynomials of degree $(m - 2)$, and so on to include 4^m constant polynomials. Moreover, most of the coefficients in these polynomials would suffer from underflow. For this reason, we only consider the first-order error terms. In particular, Algorithm 7 shows how we compute the forward error $e(z) = (p_\pi + p_\mu + p_\nu + p_\sigma)(z)$ using Horner's method applied to the polynomial whose coefficients are those of $(p_\pi + p_\mu + p_\nu + p_\sigma)$, computed using the *doubly compensated summation* method from Priest [26].

In fact, any of the summation methods from [7, 18, 26, 27, 28] that guarantee a relative error bound of 2μ can be used. For example, in [17], the authors advocate the use of the *accurate summation* method from [27, 28], the result of which is guaranteed to be *faithfully rounded*, i.e., as accurate as if computed exactly and then rounded to an adjacent floating-point number. Hence, the relative error in the accurate summation method is 2μ. However, since we only need to sum 4 complex floating-point numbers, it is more efficient to use the doubly compensated summation method. Although the result of this method is not guaranteed to be faithfully rounded, it follows from the analysis of Priest in [26, Section 4.1] that it has the same relative error of 2μ. If we ignore the cost of this sort, which for the given 4 floating-point values requires 9 absolute values, 6 comparisons, and 3 swaps, then the execution of Algorithm 8 requires 30 flops. Since Algorithm 7 uses the complex doubly compensated summation from Algorithm 9, it follows that its execution requires $68m - 8$ flops.

Algorithm 7 Horner method applied to the degree-$(m-1)$ polynomial $(p+q+r+s)$.

```
function $[v_0] = \mathrm{HornerSum}(p, q, r, s, z)$ :
  $v_{m-1} = \mathrm{DbleCompSumCmplx}(p_{m-1}, q_{m-1}, r_{m-1}, s_{m-1})$
  for $k = m-2$ to $k = 0$ do
    $v_k = \mathrm{fl}(z \cdot v_{k+1} + \mathrm{DbleCompSumCmplx}(p_k, q_k, r_k, s_k))$
  end for
```

Algorithm 8 Doubly compensated summation of $a_1, a_2, a_3, a_4 \in \mathcal{F}$.

```
function $[s_n] = \mathrm{DbleCompSum}(a_1, a_2, a_3, a_4)$ :
  Sort the $a_i$ so that $|a_1| \geq \cdots \geq |a_4|$
  $s_1 = a_1, c_1 = 0$
  for $k = 2$ to $k = 4$ do
    $y_k = \mathrm{fl}(c_{k-1} + a_k)$
    $u_k = \mathrm{fl}(a_k - (y_k - c_{k-1}))$
    $t_k = \mathrm{fl}(y_k + s_{k-1})$
    $v_k = \mathrm{fl}(y_k - (t_k - s_{k-1}))$
    $z_k = \mathrm{fl}(u_k + v_k)$
    $s_k = \mathrm{fl}(t_k + z_k)$
    $c_k = \mathrm{fl}(z_k - (s_k - t_k))$
  end for
```

Algorithm 9 Doubly compensated summation of $a_1, a_2, a_3, a_4 \in \mathcal{C}$.

```
function $[x + iy] = \mathrm{DbleCompSumCmplx}(a_1, a_2, a_3, a_4)$ :
  $x = \mathrm{DbleCompSum}(\mathrm{Re}(a_1), \mathrm{Re}(a_2), \mathrm{Re}(a_3), \mathrm{Re}(a_4))$
  $y = \mathrm{DbleCompSum}(\mathrm{Im}(a_1), \mathrm{Im}(a_2), \mathrm{Im}(a_3), \mathrm{Im}(a_4))$
```

Let $p_\pi, p_\mu, p_\nu, p_\sigma$ be the error polynomials from Algorithm 6. Then, by [17, Lemma 5.4], the result $v_0 = \mathrm{HornerSum}(p_\pi, p_\mu, p_\nu, p_\sigma, z)$ satisfies the following forward error bound:

$$|e(z) - v_0| \leq \widetilde{\gamma}_{2m-1}((p_\pi \widetilde{+ p_\mu} + p_\nu) + \widetilde{p}_\sigma)(|z|). \tag{3.3}$$

It is worth noting that [17, Lemma 5.4] is formally stated for the accurate summation method from [27, 28]; however, since the doubly compensated method has the same relative error bound, the result follows. Now, by combining the error-free transformation of the Horner method and the method for computing the corresponding forward error, we obtain the compensated Horner method shown in Algorithm 10, which requires $100m - 7$ flops to be executed.

Algorithm 10 Compensated Horner method.

```
function $[c_0] = \mathrm{CompHorner}(p, z)$ :
    $[h_0, p_\pi, p_\mu, p_\nu, p_\sigma] = \mathrm{EFTHorner}(p, z)$
    $e_0 = \mathrm{HornerSum}(p_\pi, p_\mu, p_\nu, p_\sigma, z)$
    $c_0 = \mathrm{fl}(h_0 + e_0)$
```

From [17, Theorem 5.5], we know that the result $c_0 = \mathrm{CompHorner}(p, z)$ satisfies the following forward error bound:

$$|p(z) - c_0| \leq \mu|p(z)| + \widetilde{\gamma}_{2m}^2 \widetilde{p}(|z|). \tag{3.4}$$

Furthermore, if $p(z) \neq 0$, then we have the relative forward error bound

$$\frac{|p(z) - c_0|}{|p(z)|} \leq \mu + \widetilde{\gamma}_{2m}^2 \,\mathrm{cond}(p, z).$$

Therefore, the compensated Horner method is as accurate as if computed in twice the working precision and then rounded to the working precision.

3.2. Running error bound. The error bound in (3.4) is not useful in practice since it contains the quantity $p(z)$. Therefore, we establish a running error bound that can be computed using the error polynomials produced in Algorithm 6. We begin with the following lemma.

LEMMA 3.1. *Let p, q, r, s be degree-$(m-1)$ polynomials with non-negative coefficients $a_k, b_k, c_k, d_k \in \mathcal{F}$, and let $x \in \mathcal{F}$ be non-negative. Then,*

$$0 \leq \sum_{k=0}^{m-1} (a_k + b_k + c_k + d_k)x^k \leq (1 + 2\mu)^{m-1}\,\mathrm{HornerSum}(p, q, r, s, x).$$

Proof. Consider v_0 and the intermediate values v_k produced by Algorithm 7. We prove that, for $k = 0, 1, \ldots, m-1$, we have

$$\begin{aligned} s(k) := &\sum_{j=0}^{k} (a_{m-1-k+j} + b_{m-1-k+j} + c_{m-1-k+j} + d_{m-1-k+j})x^j \\ &\leq (1 + 2\mu)^{2k+1} v_{m-1-k}. \end{aligned} \tag{3.5}$$

The base case, $k = 0$, holds since the relative error in the doubly compensated summation is bounded above by 2μ, that is,

$$\begin{aligned} (a_{m-1} + b_{m-1} + c_{m-1} + d_{m-1}) &\leq (1 + 2\mu)\,\mathrm{DbleCompSum}(a_{m-1}, b_{m-1}, c_{m-1}, d_{m-1}) \\ &= (1 + 2\mu)v_{m-1}. \end{aligned}$$

Suppose that (3.5) holds for some integer k, where $0 \leq k < m-1$, and note that $s(k+1)$ can be split into the sum of two parts:

$$\left(\sum_{j=0}^{k} (a_{m-1-k+j} + b_{m-1-k+j} + c_{m-1-k+j} + d_{m-1-k+j})x^j\right) x$$

and

$$a_{m-k-2} + b_{m-k-2} + c_{m-k-2} + d_{m-k-2}.$$

By the induction hypothesis, the first part is bounded above by $(1+2\mu)^{2k+1}x \cdot v_{m-1-k}$, and, since the relative error in the doubly compensated summation is bounded above by 2μ, the second part is bounded above by

$$(1+2\mu)\,\mathrm{DbleCompSum}(a_{m-k-2}, b_{m-k-2}, c_{m-k-2}, d_{m-k-2}).$$

Therefore, $s(k+1)$ is bounded above by

$$(1+2\mu)^{2k+1}x \cdot v_{m-1-k} + (1+2\mu)\,\mathrm{DbleCompSum}(a_{m-k-2}, b_{m-k-2}, c_{m-k-2}, d_{m-k-2}),$$

which, in turn, is bounded above by $(1+2\mu)^{2k+1}(1+\mu)^2$ times

$$\mathrm{fl}(x \cdot v_{m-1-k} + \mathrm{DbleCompSum}(a_{m-k-2}, b_{m-k-2}, c_{m-k-2}, d_{m-k-2})).$$

Thus,

$$s(k+1) \leq (1+2\mu)^{2(k+1)+1} v_{m-k-2},$$

and it follows that (3.5) holds for $k = 0, 1, \ldots, m-1$. □

We are now ready to prove a forward running error bound for the compensated Horner method in Algorithm 10.

THEOREM 3.2. *Let p be a degree-m complex polynomial as defined in (3.1). Then, $c_0 = \mathrm{CompHorner}(p, z)$ satisfies*

$$|c_0 - p(z)| \leq \mathrm{fl}\big(\mu|c_0| + (\widetilde{\gamma}_{4m+2}\,\mathrm{HornerSum}(|p_\pi|, |p_\mu|, |p_\nu|, |p_\sigma|, |z|) + 2\mu^2|c_0|)\big).$$

Proof. Note that $c_0 = \mathrm{CompHorner}(p, z)$ satisfies

$$\begin{aligned} |c_0 - p(z)| &= |\mathrm{fl}(h_0 + e_0) - p(z)| \\ &\leq |\mathrm{fl}(h_0 + e_0) - (h_0 + e_0)| + |(h_0 + e_0) - p(z)| \\ &\leq \mu|\mathrm{fl}(h_0 + e_0)| + |e_0 - e(z)|, \end{aligned}$$

where $e(z) = (p_\pi + p_\mu + p_\nu + p_\sigma)(z)$. Applying the forward error bound in (3.3) and Lemma 3.1, we have

$$\begin{aligned} |e_0 - e(z)| &\leq \widetilde{\gamma}_{2m-1}((p_\pi \widetilde{+ p_\mu +} p_\nu) + \widetilde{p}_\sigma)(|z|) \\ &\leq (1+2\mu)^{2m-1}\,\widetilde{\gamma}_{2m-1}\,\mathrm{HornerSum}(|p_\pi|, |p_\mu|, |p_\nu|, |p_\sigma|, |z|). \end{aligned}$$

Next, we make use of the following inequalities:

$$(1+\sqrt{2}\gamma_2)\widetilde{\gamma}_m \leq \widetilde{\gamma}_{m+1} \quad \text{and} \quad \widetilde{\gamma}_m \leq (1-\sqrt{2}\,\gamma_2)\widetilde{\gamma}_{m+1}.$$

The first inequality implies that $(1+\sqrt{2}\,\gamma_2)^{2m-1}\,\widetilde{\gamma}_{2m-1} \leq \widetilde{\gamma}_{4m-2}$. This combined with $2\mu \leq \sqrt{2}\,\gamma_2$ implies that $(1+2\mu)^{2m-1}\,\widetilde{\gamma}_{2m-1} \leq \widetilde{\gamma}_{4m-2}$. Furthermore, the second inequality implies that $\widetilde{\gamma}_{4m-2} \leq (1-\sqrt{2}\,\gamma_2)^4\,\widetilde{\gamma}_{4m+2}$. Since $(1-\sqrt{2}\,\gamma_2) \leq (1-\mu)^{9/4}$, it follows that $\widetilde{\gamma}_{4m-2} \leq (1-\mu)^9\widetilde{\gamma}_{4m+2}$. We summarize these results in the following inequality

$$(1+2\mu)^{2m-1}\,\widetilde{\gamma}_{2m-1} \leq \widetilde{\gamma}_{4m-2} \leq (1-\mu)^9\,\widetilde{\gamma}_{4m+2}. \tag{3.6}$$

Now, we analyze the error in the computation of

$$\widetilde{\gamma}_k = \frac{k\sqrt{2}\,\gamma_2}{1-k\sqrt{2}\,\gamma_2} = \frac{2k\mu\sqrt{2}}{(1-2\mu)-2k\mu\sqrt{2}}.$$

Note that 2μ, $2k\mu$, and $(1-2\mu)$ are floating-point numbers computed exactly, and the IEEE standard requires the square-root operation be correctly rounded, with round to nearest. Hence, $2k\mu\sqrt{2} \leq (1-\mu)^{-2}\,\mathrm{fl}\big(2k\mu\sqrt{2}\big)$ and $(1-2\mu) - 2k\mu\sqrt{2} \geq (1-\mu)^{3}\,\mathrm{fl}\big((1-2\mu) - 2k\mu\sqrt{2}\big)$, and it follows that

$$\widetilde{\gamma}_k \leq (1-\mu)^{-6}\,\mathrm{fl}(\widetilde{\gamma}_k). \tag{3.7}$$

Finally, we apply the inequalities in (3.6) and (3.7) to obtain the following:

$$\begin{aligned}|c_0 - p(z)| &\leq \mu|c_0| + (1+2\mu)^{2m-1}\,\widetilde{\gamma}_{2m-1}\,\mathrm{HornerSum}(|p_\pi|,|p_\mu|,|p_\nu|,|p_\sigma|,|z|)\\ &\leq \mu|c_0| + (1-\mu)^{9}\,\widetilde{\gamma}_{4m+2}\,\mathrm{HornerSum}(|p_\pi|,|p_\mu|,|p_\nu|,|p_\sigma|,|z|)\\ &\leq \mu|c_0| + (1-\mu)^{2}\,\mathrm{fl}(\widetilde{\gamma}_{4m+2}\,\mathrm{HornerSum}(|p_\pi|,|p_\mu|,|p_\nu|,|p_\sigma|,|z|)).\end{aligned}$$

Therefore, the forward error in the computation of c_0 is bounded above by

$$(1-\mu)\mu|c_0| + (1-\mu)^2\,\mathrm{fl}(\widetilde{\gamma}_{4m+2}\,\mathrm{HornerSum}(|p_\pi|,|p_\mu|,|p_\nu|,|p_\sigma|,|z|)) + \mu^2|c_0|,$$

which, since we can always assume that $2(1-\mu)^2 \geq 1$, is bounded above by

$$(1-\mu)\mu|c_0| + (1-\mu)^2\big(\mathrm{fl}(\widetilde{\gamma}_{4m+2}\,\mathrm{HornerSum}(|p_\pi|,|p_\mu|,|p_\nu|,|p_\sigma|,|z|)) + 2\mu^2|c_0|\big).$$

Furthermore, since we assume that no underflow occurs, both $\mu|c_0|$ and $2\mu^2|c_0|$ are floating-point numbers computed exactly, and it follows that the forward error in the computation of c_0 is bounded above by

$$(1-\mu)\big(\mu|c_0| + \mathrm{fl}\big(\widetilde{\gamma}_{4m+2}\,\mathrm{HornerSum}(|p_\pi|,|p_\mu|,|p_\nu|,|p_\sigma|,|z|) + 2\mu^2|c_0|\big)\big).$$

Therefore, we have

$$|c_0 - p(z)| \leq \mathrm{fl}\big(\mu|c_0| + (\widetilde{\gamma}_{4m+2}\,\mathrm{HornerSum}\big(|p_\pi|,|p_\mu|,|p_\nu|,|p_\sigma|,|z|\big) + 2\mu^2|c_0|)\big). \qquad \square$$

4. The Ehrlich-Aberth method.

Let p be a degree-m complex polynomial as defined in (3.1). Throughout this section, we assume that p has simple roots $\zeta_1,\ldots,\zeta_m \in \mathbb{C}$. For $1 \leq i \leq m$ and $n \geq 0$, let $z_{n,i} \in \mathcal{C}$ denote the approximation of ζ_i after n iterations. The Ehrlich-Aberth method [1, 3, 4, 8] updates each root approximation as follows:

$$z_{n+1,i} = \mathrm{fl}\left(z_{n,i} - \frac{p(z_{n,i})}{p'(z_{n,i}) - p(z_{n,i})A_{n,i}(z_{n,i})}\right), \tag{4.1}$$

where $A_{n,i}(z) = \prod_{j\neq i}(z - z_{n,j})^{-1}$. Note that we refer to the fractional expression on the right-hand side of (4.1) as the *Ehrlich-Aberth correction term*.

The polynomial evaluations in (4.1) can be computed using the Horner method from Algorithm 5. Moreover, given $z \in \mathcal{C}$ and $h_0 = \mathrm{Horner}(p,z)$, we can push the error in the computation back to the coefficients as follows [3, Theorem 7]:

$$h_0 = \sum_{i=0}^{m} a_i(1+\epsilon_i)z^i,$$

where $|\epsilon_i| \leq \big((2\sqrt{2}+1)i + 1\big)\mu + O(\mu^2)$. Note that the backward error of z as a root of the polynomial p is defined by

$$\eta(z) = \min\{\epsilon\colon (p+\Delta p)(z) = 0,\ |\Delta a_i| \leq \epsilon|e_i|,\ i = 0,1,\ldots,m\},$$

where the e_i are arbitrary and represent tolerances against which perturbations are measured, and

$$\Delta p(z) = \Delta a_m z^m + \cdots + \Delta a_1 z + \Delta a_0.$$

It is well-known (see, e.g., [5]) that

$$\eta(z) = \frac{|p(z)|}{\alpha(|z|)},$$

where $\alpha(z) = \sum_{i=0}^{m} |e_i| z^i$. Let $e_i = ((2\sqrt{2}+1)i+1)a_i$. Then, the inequality

$$\eta(z) \le \mu \tag{4.2}$$

guarantees that z is a root of $(p + \Delta p)$, where $|\Delta a_i|$ is not larger than the upper bound of $|a_i|\epsilon_i$, for all $i = 0, 1, \ldots, m$. Furthermore, if $\eta(z) > \mu$, then z being a root of $(p + \Delta p)$ implies that $|\Delta a_i|$ is larger than $|a_i|\epsilon_i$, for some $i = 0, 1, \ldots, m$. Hence, in exact arithmetic, (4.2) denotes a stopping criterion that guarantees that iterations do not terminate until z is a root of a polynomial whose coefficients are no more perturbed than the floating-point computation of $p(z)$. In practice, we stop updating the root approximation z if the following inequality holds:

$$|h_0| > \mu \, \mathrm{Horner}(\alpha, |z|), \tag{4.3}$$

where the coefficients of α have been stored as floating-point numbers.

4.1. The compensated Ehrlich-Aberth method. Let $z \in \mathcal{C}$ be a root approximation of the polynomial p such that the stopping criterion (4.3) holds. Then, $h_0 = \mathrm{Horner}(p, z)$ is no longer a reliable computation for updating the root approximation z. However, the computation $c_0 = \mathrm{CompHorner}(p, z)$ may still be reliable, depending on the value of $\mathrm{cond}(p, z)$ as defined in (3.2). In this case, we update the root approximation using (4.1) and the compensated Horner method. If the polynomial has well-conditioned simple roots, then we only need to apply the compensated Horner method to p. However, the more difficult polynomial equations have multiple or near multiple roots, which may be ill-conditioned. For this reason, we use the compensated Horner method to evaluate p and p'. A method for performing these evaluations is derived from [19, Algorithm 2] and presented in Algorithm 11.

It is clear that the value of c_0 from Algorithm 11 is the same as its value from Algorithm 10; therefore, c_0 satisfies the forward error bound in (3.4) and the running error bound in Theorem 3.2. Moreover, the derivative value c_0' can be shown to satisfy a similar forward error bound by appealing to [19, Theorem 4] and making the necessary changes from real to complex floating-point arithmetic.

Furthermore, the value of s_0 from Algorithm 11 satisfies

$$s_0 = \mathrm{HornerSum}(|p_\pi|, |p_\mu|, |p_\nu|, |p_\sigma|, |z|),$$

where p_π, p_μ, p_ν, and p_σ are the polynomials created by $\mathrm{EFTHorner}(p, z)$. Therefore, by Theorem 3.2, it follows that

$$|c_0 - p(z)| \le \mathrm{fl}\big(\mu|c_0| + (\tilde{\gamma}_{4m+2} s_0 + 2\mu^2 |c_0|)\big).$$

Hence, if

$$\mathrm{fl}\big(\mu|c_0| + (\tilde{\gamma}_{4m+2} s_0 + 2\mu^2 |c_0|)\big) < |c_0|,$$

Algorithm 11 Compensated Horner method for p and p'.

```
function $[c_0, c_0', s_0] = \mathrm{CompHornerDer}(p, z)$ :
    $h_m = a_m$, $h_m' = 0$
    $e_m = 0$, $e_m' = 0$
    $s_m = 0$
    **for** $k = m - 1$ **to** $k = 0$ **do**
        $[\hat{h}_k', \pi_k', \mu_k', \nu_k'] = \mathrm{TwoProductCmplx}\big(h_{k+1}', z\big)$
        $[h_k', \sigma_k'] = \mathrm{TwoSumCmplx}\Big(\hat{h}_k', h_{k+1}\Big)$
        $e_k' = \mathrm{fl}\big(z \cdot e_{k+1}' + e_{k+1} + \mathrm{DbleCompSumCmplx}(\pi_k', \mu_k', \nu_k', \sigma_k')\big)$
        $[\hat{h}_k, \pi_k, \mu_k, \nu_k] = \mathrm{TwoProductCmplx}(h_{k+1}, z)$
        $[h_k, \sigma_k] = \mathrm{TwoSumCmplx}\Big(\hat{h}_k, a_k\Big)$
        $e_k = \mathrm{fl}(z \cdot e_{k+1} + \mathrm{DbleCompSumCmplx}(\pi_k, \mu_k, \nu_k, \sigma_k))$
        $s_k = \mathrm{fl}(|z| \cdot s_{k+1} + \mathrm{DbleCompSum}(|\pi_k|, |\mu_k|, |\nu_k|, |\sigma_k|))$
    **end for**
    $c_0 = \mathrm{fl}(h_0 + e_0)$, $c_0' = \mathrm{fl}(h_0' + e_0')$
```

then the error in the computation of $p(z)$ is smaller than the size of $|c_0|$ and it is still a reliable computation. Thus, we obtain the following stopping criterion for the compensated Ehrlich-Aberth method:

$$|c_0| \le \mathrm{fl}\big(\mu|c_0| + (\tilde{\gamma}_{4m+2}s_0 + 2\mu^2|c_0|)\big). \tag{4.4}$$

Finally, let $z \in \mathcal{C}$ be an approximate root of the polynomial p, and consider the computation $[c_0, c_0', s_0] = \mathrm{CompHornerDer}(p, z)$. Denote by $EA(p, z)$ the computed Ehrlich-Aberth correction term using the values of c_0 and c_0'. Then, even if (4.4) does not hold, it may happen that the following inequality does hold,

$$|EA(p, z)| \le \mu|z|, \tag{4.5}$$

which implies that the relative change made to z by the Ehrlich-Aberth correction term is insignificant. In summary, we do not stop updating the root approximations from the compensated Ehrlich-Aberth method until either (4.4) or (4.5) hold, which guarantees that the updates do not cease until the relative error in the compensated Horner method is too large or the relative size of the Ehrlich-Aberth correct term is too small.

Before proceeding, note that the Horner and compensated Horner methods are prone to overflow; for instance, when a large-degree polynomial with positive coefficients is evaluated at z, where $|z| > 1$. For this reason, the reversal polynomial $p_R(z) := z^m p(1/z)$, which for $\rho = 1/z$ satisfies

$$\begin{aligned} p(z) &= z^m p_R(\rho) \\ p'(z) &= mz^{m-1}p_R(\rho) - z^{m-2}p_R'(\rho), \end{aligned}$$

is used to compute the Ehrlich-Aberth correction term when $|z| > 1$ [3].

4.2. Limiting accuracy.

In [29], Tisseur examined the multi-variable Newton method, allowing for extended precision in the computation of the residual and its application to iterative refinements for the generalized eigenvalue problem. In particular, for a close enough eigenvalue approximation, Newton's method converges to an eigenvalue approximation with limiting accuracy that depends on the relative rounding error μ and the accuracy of the computed residual.

Then, in [14], Graillat applied this result to real polynomials with real simple zeros. Specifically, if the polynomial evaluation is computed using the real compensated Horner method, then for a close enough root approximation, Newton's method converges to a root approximation with limiting accuracy as if computed in twice the working precision and then rounded to the working precision. In this section, we show that the compensated Ehrlich-Aberth method has a similar limiting accuracy for all, real or complex, roots of a polynomial.

To this end, we rewrite (4.1) as follows:

$$z_{n+1,i} = z_{n,i} - \left(J_{n,i}(z_{n,i}) + e'_{n,i}\right)^{-1}(p(z_{n,i}) + e_{n,i}) + \epsilon_{n,i}, \tag{4.6}$$

where $J_{n,i}(z) = p'(z) - p(z)A_{n,i}(z)$, $e_{n,i}$ is the error in computing $p(z_{n,i})$, $e'_{n,i}$ is the error in computing $J_{n,i}(z_{n,i})$ and performing the division, and $\epsilon_{n,i}$ is the error in the subtraction.

For $i = 1, \ldots, m$, we assume that $p(z_{n,i})$ is computed via the compensated Horner method in Algorithm 10. Hence,

$$|e_{n,i}| \leq \mu|p(z_{n,i})| + \widetilde{\gamma}_{2m}^2\widetilde{p}(|z_{n,i}|).$$

We also assume that

$$\left|e'_{n,i}\right| \leq \mu\phi(p, z_{n,i}, m, \mu),$$

for some function ϕ that reflects the error in computing $J_{n,i}(z_{n,i})$ and performing the division. For the error $\epsilon_{n,i}$, we have

$$|\epsilon_{n,i}| \leq \mu\left(|z_{n,i}| + \left|\left(J_{n,i}(z_{n,i}) + e'_{n,i}\right)^{-1}(p(z_{n,i}) + e_{n,i})\right|\right).$$

Since the roots of p, denoted by $\zeta_1, \ldots, \zeta_m$, are assumed to be simple, it follows that for $i = 1, \ldots, m$, there exists a closed disk D_i in the complex plane centered at ζ_i such that $\zeta_j \notin D_i$, for all $j \neq i$. Furthermore, for close enough root approximations, we can assume that $z_{n,i} \in D_i$ and $z_{n,j} \notin D_i$, for all $j \neq i$. Under these assumptions, we have the following result.

LEMMA 4.1. *For $i = 1, \ldots, m$, there exists a $\beta_i > 0$ such that for all $z, w \in D_i$, we have*

$$|p(z) - p(w) - J_{n,i}(w)(z-w)| \leq \frac{\beta_i}{2}|z-w|^2 + |p(w)||A_{n,i}(w)||z-w|.$$

Proof. Since p' is Lipschitz continuous on D_i, there exists a $\beta_i > 0$ such that

$$|p'(z) - p'(w)| \leq \beta_i|z-w|,$$

for all $z, w \in D_i$. Furthermore, for all $z, w \in D_i$, $p(z) - p(w) - J_{n,i}(w)(z-w)$ can be written as

$$\int_0^1 (p'(w + t(z-w)) - p'(w))(z-w)dt + p(w)A_{n,i}(w)(z-w).$$

Therefore, by Hölder's inequality and the Lipschitz continuity of p', we have

$$|p(z) - p(w) - J_{n,i}(w)(z-w)| \leq \frac{\beta_i}{2}|z-w|^2 + |p(w)||A_{n,i}(w)||z-w|. \qquad \square$$

From Lemma 4.1 and the continuity of p and $A_{n,i}$ on D_i, it follows that there exists $M_i > 0$ such that

$$|p(z) - p(w) - J_{n,i}(w)(z-w)| \leq \frac{\beta_i}{2}|z-w|^2 + M_i|z-w|,$$

for all $z, w \in D_i$. Furthermore, the Lipschitz continuity of $J_{n,i}$ on D_i implies that there exists $\beta_i' > 0$ such that

$$|J_{n,i}(z) - J_{n,i}(w)| \leq \beta_i'|z-w|,$$

for all $z, w \in D_i$.

THEOREM 4.2. *For $i = 1, \ldots, m$, assume that*

$$\left|J_{n,i}(z_{n,i})^{-1}e'_{n,i}\right| \leq \nu < 1, \tag{4.7}$$

and

$$\beta_i'\left|J_{n,i}(\zeta_i)^{-1}\right||z_{n,i} - \zeta_i| \leq \tau < 1. \tag{4.8}$$

Then, $z_{n+1,i}$ in (4.6) is well-defined and satisfies

$$|z_{n+1,i} - \zeta_i| \leq G_i|z_{n,i} - \zeta_i| + g_i,$$

where

$$G_i = \frac{\nu}{1-\nu} + \frac{\mu(2+\mu)}{(1-\tau)(1-\nu)} + M_i + \mu + \frac{(1+\mu)^2\tau}{2(1-\tau)(1-\nu)}$$

and

$$g_i = \mu|\zeta_i| + \widetilde{\gamma}_{2m}^2\frac{1+\mu}{(1-\tau)(1-\nu)}\frac{\widetilde{p}(|z_{n,i}|)}{|p'(\zeta_i)|}.$$

Proof. By (4.8) and the Lipschitz continuity of $J_{n,i}$ on D_i, we have

$$\left|J_{n,i}(\zeta_i)^{-1}\right||J_{n,i}(z_{n,i}) - J_{n,i}(\zeta_i)| \leq \beta_i'\left|J_{n,i}(\zeta_i)^{-1}\right||z_{n,i} - \zeta_i| \leq \tau < 1. \tag{4.9}$$

Also, note that

$$J_{n,i}(z_{n,i})^{-1} = \left(1 + \frac{J_{n,i}(z_{n,i}) - J_{n,i}(\zeta_i)}{J_{n,i}(\zeta_i)}\right)^{-1} J_{n,i}(\zeta_i)^{-1},$$

which, combined with (4.9), implies that

$$\left|J_{n,i}(z_{n,i})^{-1}\right| \leq \frac{\left|J_{n,i}(\zeta_i)^{-1}\right|}{1 - |J_{n,i}(\zeta_i)^{-1}||J_{n,i}(z_{n,i}) - J_{n,i}(\zeta_i)|} \leq \frac{\left|J_{n,i}(\zeta_i)^{-1}\right|}{1-\tau}. \tag{4.10}$$

Similarly, (4.7) and (4.10) give us

$$\left|\left(J_{n,i}(z_{n,i}) + e'_{n,i}\right)^{-1}\right| \leq \frac{\left|J_{n,i}(z_{n,i})^{-1}\right|}{1 - \left|J_{n,i}(z_{n,i})^{-1}e'_{n,i}\right|} \leq \frac{\left|J_{n,i}(\zeta_i)^{-1}\right|}{(1-\tau)(1-\nu)}.$$

Therefore, $\left(J_{n,i}(z_{n,i}) + e'_{n,i}\right)$ is non-zero, so $z_{n+1,i}$ in (4.6) is well-defined and

$$\begin{aligned} z_{n+1,i} - \zeta_i &= z_{n,i} - \zeta_i - \left(J_{n,i}(z_{n,i}) + e'_{n,i}\right)^{-1}(p(z_{n,i}) + e_{n,i}) + \epsilon_{n,i} \\ &= \left(1 - \frac{J_{n,i}(z_{n,i})}{J_{n,i}(z_{n,i}) + e'_{n,i}}\right)(z_{n,i} - \zeta_i) - \frac{p(z_{n,i}) - J_{n,i}(z_{n,i})(z_{n,i} - \zeta_i) + e_{n,i}}{J_{n,i}(z_{n,i}) + e'_{n,i}} + \epsilon_{n,i}. \end{aligned}$$

Hence, we have the bound

$$\begin{aligned} |z_{n+1,i} - \zeta_i| \leq &\left|1 - \frac{J_{n,i}(z_{n,i})}{J_{n,i}(z_{n,i}) + e'_{n,i}}\right| |z_{n,i} - \zeta_i| \\ &+ \frac{|p(z_{n,i}) - J_{n,i}(z_{n,i})(z_{n,i} - \zeta_i)| + |e_{n,i}|}{\left|J_{n,i}(z_{n,i}) + e'_{n,i}\right|} + |\epsilon_{n,i}|. \end{aligned}$$

From

$$1 - \frac{J_{n,i}(z_{n,i})}{J_{n,i}(z_{n,i}) + e'_{n,i}} = \frac{e'_{n,i}}{J_{n,i}(z_{n,i}) + e'_{n,i}} = \frac{J_{n,i}(z_{n,i})^{-1}e'_{n,i}}{1 + J_{n,i}(z_{n,i})^{-1}e'_{n,i}}$$

and (4.7), we have

$$\left|1 - \frac{J_{n,i}(z_{n,i})}{J_{n,i}(z_{n,i}) + e'_{n,i}}\right| \leq \frac{\nu}{1-\nu}.$$

From Lemma 4.1, we have

$$|p(z_{n,i}) - J_{n,i}(z_{n,i})(z_{n,i} - \zeta_i)| \leq \frac{\beta_i}{2}|z_{n,i} - \zeta_i|^2 + M_i|z_{n,i} - \zeta_i|$$

and

$$|p(z_{n,i}) - J_{n,i}(\zeta_{n,i})(z_{n,i} - \zeta_i)| \leq \frac{\beta_i}{2}|z_{n,i} - \zeta_i|^2.$$

Hence,

$$\begin{aligned} |p(z_{n,i})| &\leq |p(z_{n,i}) - J_{n,i}(\zeta_i)(z_{n,i} - \zeta_i)| + |J_{n,i}(\zeta_i)(z_{n,i} - \zeta_i)| \\ &\leq \frac{\beta_i}{2}|z_{n,i} - \zeta_i|^2 + |J_{n,i}(\zeta_i)||z_{n,i} - \zeta_i|, \end{aligned}$$

which allows us to bound the error term $e_{n,i}$ as

$$|e_{n,i}| \leq \mu\left(\frac{\beta_i}{2}|z_{n,i} - \zeta_i|^2 + |J_{n,i}(\zeta_i)||z_{n,i} - \zeta_i|\right) + \widetilde{\gamma}_{2m}^2\widetilde{p}(|z_{n,i}|)$$

and the error term $\epsilon_{n,i}$ as

$$|\epsilon_{n,i}| \leq \mu(|z_{n,i} - \zeta_i| + |\zeta_i| + |d_{n,i}|),$$

where

$$\begin{aligned} |d_{n,i}| &= \left|\left(J_{n,i}(z_{n,i}) + e'_{n,i}\right)^{-1}\right|(|p(z_{n,i})| + |e_{n,i}|) \\ &\leq \frac{\left|J_{n,i}(\zeta_i)^{-1}\right|}{(1-\tau)(1-\nu)}\left((1+\mu)\left(\frac{\beta_i}{2}|z_{n,i} - \zeta_i|^2 + |J_{n,i}(\zeta_i)||z_{n,i} - \zeta_i|\right) + \widetilde{\gamma}_{2m}^2\widetilde{p}(|z_{n,i}|)\right). \end{aligned}$$

□

Note that conditions (4.7) and (4.8) are necessary for $z_{n+1,i}$ in (4.6) to be defined. Assumption (4.7) is a condition on the accuracy of computing $J_{n,i}(z)$, and assumption (4.8) is a condition on the diameter of the disk D_i and the conditioning of the root ζ_i. In our final result, we assume that we can make ν, τ, and M_i small enough, say $\nu \leq \frac{1}{8}$, $\tau \leq \frac{1}{8}$, and $M_i \leq \frac{1}{8}$. Under these assumptions, we have the following result, which implies that under suitable conditions the compensated Ehrlich-Aberth method computes all roots of a polynomial as accurate as if computed in twice the working precision and then rounded to the working precision.

COROLLARY 4.3. *For $i = 1, \ldots, m$, let $z_{0,i} \in D_i$. Then, the compensated Ehrlich-Aberth method generates a sequence of approximations whose absolute error decreases until the first n for which*

$$|z_{n+1,i} - \zeta_i| \approx \mu|\zeta_i| + \widetilde{\gamma}_{2m}^2 \frac{\widetilde{p}(|\zeta_i|)}{|p'(\zeta_i)|}. \tag{4.11}$$

Furthermore, if $\zeta_i \neq 0$, then we have the relative limiting accuracy

$$\frac{|z_{n+1,i} - \zeta_i|}{|\zeta_i|} \approx \mu + \widetilde{\gamma}_{2m}^2 \operatorname{cond}(p, \zeta_i),$$

where

$$\operatorname{cond}(p, \zeta) = \frac{\widetilde{p}(|\zeta|)}{|\zeta||p'(\zeta)|}$$

is the condition number for the computation of the root ζ of p.

Proof. By Theorem 4.2, for $i = 1, \ldots, m$, we have

$$|z_{1,i} - \zeta_i| \leq G_i|z_{0,i} - \zeta_i| + g_i.$$

Under the assumption that $\nu \leq \frac{1}{8}$, $\tau \leq \frac{1}{8}$, and $M_i \leq \frac{1}{8}$, we have $G_i \leq \frac{1}{2}$, and it follows that the absolute error contracts unless (4.11) already holds. Thus, the result follows via induction. □

5. Numerical experiments.

In this section, we present the results of several numerical experiments to demonstrate the running error bound of the compensated Horner method in Theorem 3.2 and the limiting accuracy of the compensated Ehrlich-Aberth method in Corollary 4.3. In addition, we compare the computation time of the compensated Horner method to the Horner method implemented in double and quadruple precision. Finally, we compare the computation time, backward error, and forward error of the compensated Ehrlich-Aberth method to the Ehrlich-Aberth method implemented in quadruple precision. Note that all higher precision computations, such as those in quadruple precision, are implemented using the GNU MPFR and MPC libraries [9, 10]. The results that follow are from tests run on an Intel Core i7 CPU running 3.2GHz with 16GB of memory. All code is written in C and compiled using Apple clang version 12.0.5 and is available at https://github.com/trcameron/CompEA.

5.1. The compensated Horner method.

In this section, we illustrate the running error bound of the compensated Horner method in Theorem 3.2 and the computation time of the compensated Horner method in comparison to the Horner method implemented in double and quadruple precision.

5.1.1. Running error bound. The numerical experiment is designed as follows. We evaluate the expanded form of $p(z) = (z - (1 + i))^5$ for 2000 z-values near the root $1 + i$. For each value of z, we compute $\mathrm{CompHorner}(p, z)$, the running error bound in Theorem 3.2, and the a priori error bound in (3.4). Then, we compute an accurate evaluation of $p(z)$ in a high precision format, which allows us to measure the forward error in the computation $\mathrm{CompHorner}(p, z)$. The results are displayed in Figure 5.1, where $\mathrm{Im}(z) = 1$ and $\mathrm{Re}(z)$ ranges from 0.99 to 1.01 with a step size of 10^{-5}. Note that as z gets closer to the root $1 + i$, the condition number increases, and the a priori error bound becomes more pessimistic. The running error bound is more accurate as it takes into account the rounding errors that occur during the computation.

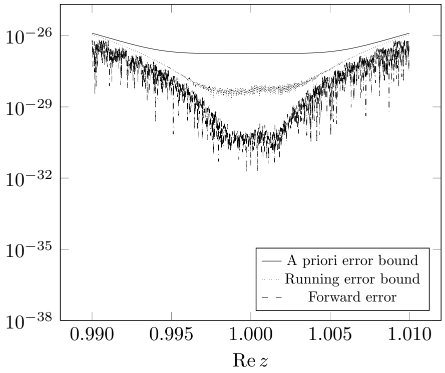

FIG. 5.1. *Running error bound test.*

5.1.2. Computation time. In addition to comparing the computation time of the compensated Horner method to the Horner method implemented in double and quadruple precision, we compare the computation time of the compensated Horner method when the fast accurate summation method from [28] is used versus the doubly compensated summation method in Algorithm 7. Note that this test illustrates our point made in Section 3.1 that the doubly compensated summation method is more efficient in this case.

The experiment is designed as follows. For each degree from 100 to 2500 incremented by 10, we form 100 random complex polynomials. For each polynomial, the computation time required to evaluate the polynomial at a random complex number is recorded. Then, the average time for each degree is displayed in Figure 5.2. Note that the compensated Horner method is about 2 times faster with the doubly compensated summation method versus with the fast accurate summation method. Moreover, the compensated Horner method with the doubly compensated summation method is about 4 times faster than the Horner method implemented in quadruple precision.

5.2. The compensated Ehrlich-Aberth method. In this section, we illustrate the limiting accuracy of the compensated Ehrlich-Aberth method in Corollary 4.3. In addition, we compare the accuracy and computation time of the compensated Ehrlich-Aberth method to the Ehrlich-Aberth method implemented in double and quadruple precision. Finally, the experiments in this section illustrate the robustness of the stopping criteria for the compensated Ehrlich-Aberth method discussed in Section 4.1. Indeed, in all experiments, the stopping criteria were used to suspend the compensated Ehrlich-Aberth iterations, and, as evident by the accuracy of the root approximations, these iterations were not suspended prematurely.

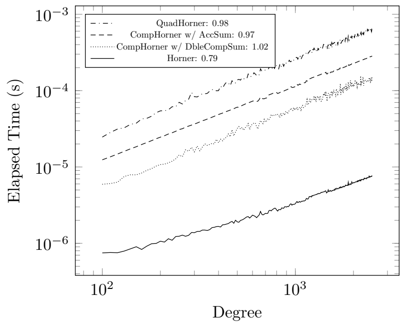

FIG. 5.2. *Computation time test.*

5.2.1. Limiting accuracy. The experiment is designed as follows. Chebyshev polynomials of the first kind are created for each degree from 5 to 80, incremented by 1. It is known that the roots of the Chebyshev polynomial become more ill-conditioned as the degree of the polynomial increases. In Figure 5.3, we compare the maximum condition number of the roots of each Chebyshev polynomial to the maximum forward error in the computed roots using the Ehrlich-Aberth method implemented in double precision and the compensated Ehrlich-Aberth method. In addition, we include the limiting accuracy bounds for both methods.

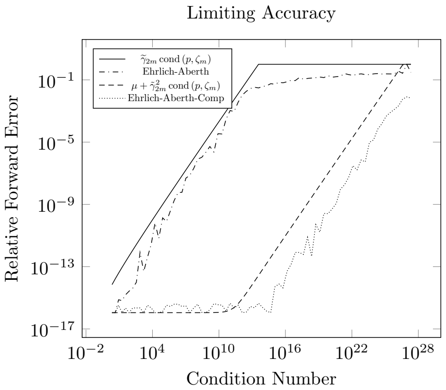

FIG. 5.3. *Limiting accuracy test.*

5.2.2. Computation time. In this section, we illustrate that the compensated Ehrlich-Aberth method is approximately 10 times faster than the quadruple precision implementation of the Ehrlich-Aberth method. Note that the latter method used the double precision implementation of the Ehrlich-Aberth method for initial estimates and then updates those estimates in quadruple precision. Hence, the number of iterations that the latter method performs in quadruple precision is comparable to the number of iterations that the former method uses the compensated Horner method. Moreover, we show that the backward error of the root approximations from these two methods is comparable.

Throughout this section, we let *Ehrlich-Aberth* and *Ehrlich-Aberth-Quad* denote the Ehrlich-Aberth method implemented in double and quadruple precision, respectively. In addition, we let *Ehrlich-Aberth-Comp* denote the compensated Ehrlich-Aberth method. Finally, we make use of the following measurement of the backward error. Given a set of root approximations $z_1, \ldots, z_m$ for the polynomial p, we define the relative backward error of all root approximations by

$$\eta(z_1, \ldots, z_m) = \frac{\|p - q\|_\infty}{\|p\|_\infty},$$

where $\|\cdot\|_\infty$ denotes the infinity vector norm and q is the polynomial with leading coefficient equal to the leading coefficient of p and whose roots are exactly $z_1, \ldots, z_m$. Note that we compute the coefficients of the polynomial q in a high precision format using the GNU MPC and MPFR libraries [9, 10].

First, we consider polynomials with well-conditioned roots. In particular, for the degrees $m = 10, 20, 40, \ldots, 2560$, we create 10 polynomials with random complex coefficients that have real and imaginary parts distributed in the interval $[-1, 1]$. We compute the roots of each polynomial using *Ehrlich-Aberth*, *Ehrlich-Aberth-Quad*, and *Ehrlich-Aberth-Comp*, and the relative backward error and the elapsed computation time is recorded. Then, the average over all 10 polynomials for each degree is recorded on the left of Figure 5.4. On the right of Figure 5.4, the same test is repeated for polynomials of the form

$$p(z) = 1 + 2z + \cdots + (m + 1)z^m,$$

for the degrees $m = 10, 20, 40, \ldots, 2560$.

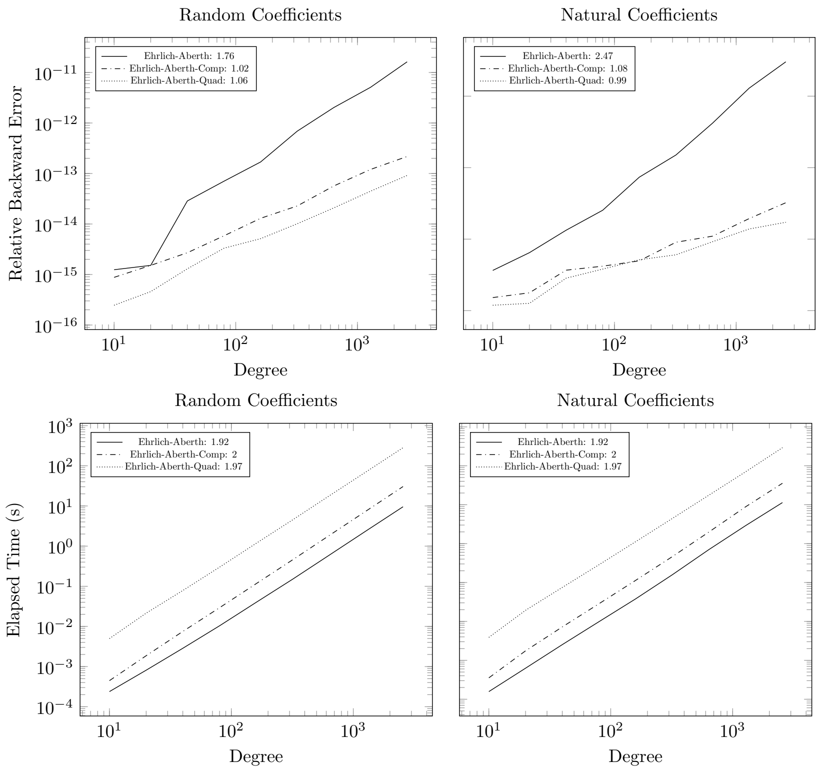

FIG. 5.4. *Well-conditioned polynomial roots test.*

Note that the slope of the linear regression line is reported for both the backward error and the elapsed time in Figure 5.4. In particular, *Ehrlich-Aberth-Quad* is approximately 25 times slower than *Ehrlich-Aberth*, whereas, *Ehrlich-Aberth-Comp* is only 2–3 times slower. Moreover, the backward error of the *Ehrlich-Aberth-Quad* and the *Ehrlich-Aberth-Comp* method is comparable.

Next, we consider polynomials with ill-conditioned roots. In particular, for the degrees $m = 10, 20, 30, \ldots, 150$, we create 10 polynomials with random complex roots on the unit circle. Then, we compute their roots using *Ehrlich-Aberth*, *Ehrlich-Aberth-Quad*, and *Ehrlich-Aberth-Comp*, and the relative backward error and the elapsed computation time is recorded. The average over all 10 polynomials for each degree is recorded on the left of Figure 5.5. On the right of Figure 5.5, the same test is repeated for the truncated exponential:

$$p(z) = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \cdots + \frac{1}{m!}x^m,$$

for the degrees $m = 10, 20, 30, \ldots, 100$.

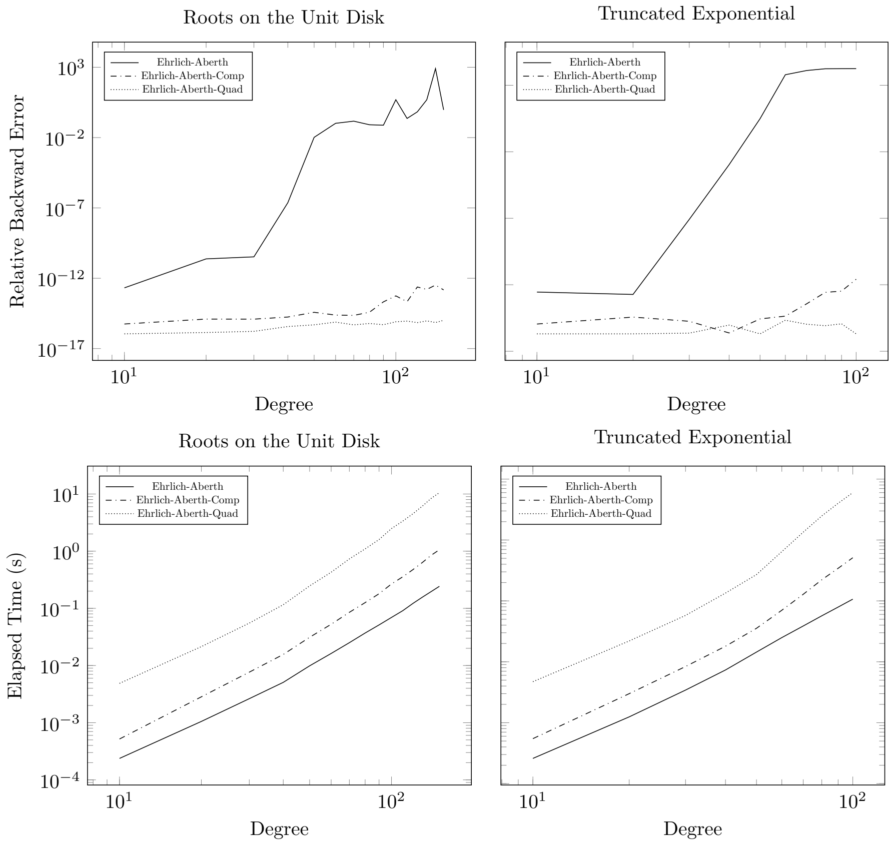

FIG. 5.5. *Ill-conditioned polynomial roots test.*

As with the well-conditioned polynomial roots test, the *Ehrlich-Aberth-Comp* method is approximately 10 times faster than the *Ehrlich-Aberth-Quad* method in the ill-conditioned polynomial roots test. Note that the polynomial roots become more ill-conditioned as the degree of the polynomial increases, which causes the accuracy of the *Ehrlich-Aberth* method to rapidly decline. In contrast, *Ehrlich-Aberth-Comp* is able to reduce the relative backward error of all computed roots to $O(\mu)$ for polynomials of degree less than 80. Of course, once the condition number of the polynomial roots is too large, say greater than 10^{16}, the additional accuracy afforded by the compensated Horner method is lost.

5.2.3. Forward error. In this section, we illustrate the improvements made to the forward error of root approximations using the compensated Ehrlich-Aberth method developed in Section 4.1. In particular, we investigate polynomials with simple ill-conditioned roots and polynomials with multiple and near multiple roots. Throughout this section, we let *Ehrlich-Aberth* and *Ehrlich-Aberth-Quad* denote the Ehrlich-Aberth method implemented in double and quadruple precision, respectively. In addition, we let *Ehrlich-Aberth-Comp* denote the compensated Ehrlich-Aberth method.

First, we consider polynomials with simple ill-conditioned roots. In particular, for the degrees $m = 5, 6, \ldots, 20$, we consider the monic mth-degree polynomial with prescribed roots

$$2^{-\lfloor m/2 \rfloor + j} - 3,$$

for $j = 0, 1, \ldots, m-1$, where $\lfloor \cdot \rfloor$ denotes the floor function. For each polynomial, the maximum relative forward error of the root approximations is recorded on the left of Figure 5.6. Next, for the degrees $m = 5, 6, \ldots, 25$, we consider monic mth-degree polynomials whose roots have small imaginary parts:

$$j + (-1)^j 8\mu i,$$

for $j = 1, 2, \ldots, m$. For each polynomial, the maximum relative forward error of the root approximations is recorded in the center plot of Figure 5.6. Finally, we consider Wilkinson polynomials of degrees $m = 5, 6, \ldots, 20$ [30]. For each polynomial, the maximum relative forward error of the root approximations is recorded on the right of Figure 5.6.

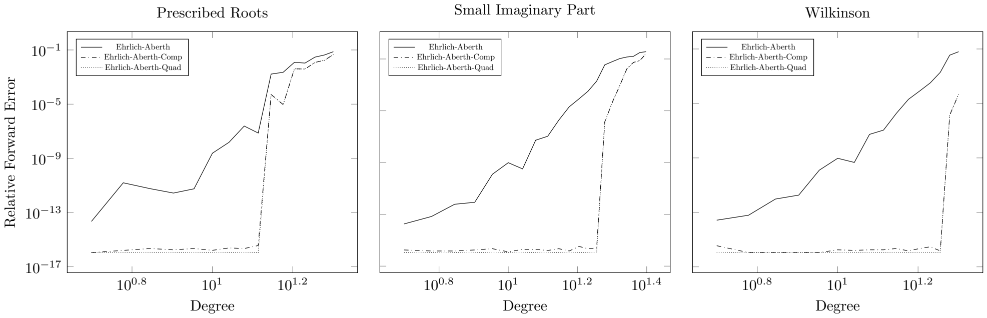

FIG. 5.6. *Simple ill-conditioned roots test.*

Note that the results in Figure 5.6 illustrate the limiting accuracy of the compensated Ehrlich-Aberth method indicated by Corollary 4.3. That is, the root approximations in these examples have a relative forward error as if they were computed in twice the working precision and then rounded to the working precision. Moreover, in each of the tests, we can see the point at which the maximum condition number of the polynomial roots became too large and we lose the additional pseudo-precision afforded to us by the compensated Horner method.

Next, we consider polynomials with multiple or near multiple roots. In particular, the Mandelbrot polynomial, which has roots that lie in a fractal (Mandelbrot set), is defined recursively by $p_0(z) = 1$ and

$$p_j(z) = z p_{j-1}^2 + 1,$$

TABLE 5.1
Multiple and near multiple roots test.

Polynomial	Limiting Accuracy	*Ehrlich-Aberth*	*Ehrlich-Aberth-Comp*	*Ehrlich-Aberth-Quad*
Mandelbrot	$2.53 \cdot 10^{-5}$	$4.42 \cdot 10^{-1}$	$3.04 \cdot 10^{-8}$	$1.70 \cdot 10^{-11}$
Kameny, $c = 10$	$1.11 \cdot 10^{-16}$	$1.42 \cdot 10^{-14}$	$1.77 \cdot 10^{-16}$	$1.11 \cdot 10^{-16}$
Kameny, $c = 10^3$	$1.11 \cdot 10^{-16}$	$5.11 \cdot 10^{-11}$	$1.25 \cdot 10^{-16}$	$1.11 \cdot 10^{-16}$
Mignotte, $a = 10$, $k = 4, m = 15$	$1.74 \cdot 10^{-16}$	$2.14 \cdot 10^{-5}$	$2.18 \cdot 10^{-16}$	$1.11 \cdot 10^{-16}$
$p_1(z)$	N/A	$3.76 \cdot 10^{-3}$	$3.02 \cdot 10^{-6}$	$8.88 \cdot 10^{-7}$
$p_2(z)$	N/A	$7.19 \cdot 10^{-4}$	$8.40 \cdot 10^{-8}$	$1.90 \cdot 10^{-8}$
$p_3(z)$	N/A	$1.56 \cdot 10^{-2}$	$7.86 \cdot 10^{-8}$	$1.04 \cdot 10^{-8}$

for $j = 1, 2, \dots, k$, with degree $m = 2^k - 1$. For our purposes, we consider $m = 63$, i.e., $k = 6$. Also, we consider the Kameny polynomial, which is defined by

$$p(z) = (c^2 z^2 - 3)^2 + c^2 z^9.$$

For $c = 10, 10^3$, these polynomials have two close real roots with 3 and 10 common decimal digits, respectively, and a complex pair with very small imaginary parts, 10^{-4} and 10^{-13}, respectively. In addition, we consider the Mignotte polynomials defined as follows:

$$p(z) = z^m + (a \cdot z - 1)^k,$$

where $a > 1$ and k is a natural number. These polynomials have a cluster of k zeros around $1/a$. Finally, we consider the following polynomials with multiple roots:

$$\begin{aligned} p_1(z) &= (z+1)^5 (z^{50} + z + 1), \\ p_2(z) &= (z-1)^4 (z^2 + z + 5)^3 (3z - 1)^2 (z^{50} + 1), \\ p_3(z) &= (z - 15)^2 \prod_{i=1}^{15} (z - i). \end{aligned}$$

Note that the non-multiple roots of $p_1(z)$ and $p_2(z)$ are well-conditioned, whereas the non-multiple roots of $p_3(z)$ are ill-conditioned.

In Table 5.1, we display the maximum relative forward error in the root computations via *Ehrlich-Aberth*, *Ehrlich-Aberth-Quad*, and *Ehrlich-Aberth-Comp*. In addition, the worst case limiting accuracy is displayed for each polynomial to illustrate the result in Corollary 4.3. Finally, note that this test illustrates the importance of using the compensated Horner method to evaluate the polynomial derivative. Indeed, if the Horner method implemented in double precision is used, then the forward error of *Ehrlich-Aberth-Comp* is $2.42 \cdot 10^{-1}$ for the Mandelbrot polynomial. Hence, in this case, the limiting accuracy is not attained unless the compensated Horner method is used to evaluate the polynomial and its derivative.

6. Conclusion. The compensated Ehrlich-Aberth method is effective for the accurate computation of all roots, real or complex, of a polynomial. In Theorem 3.2, we proved a running error bound for the compensated Horner method. Then, in Section 4.1, we used this error bound to form robust stopping criteria for the compensated Ehrlich-Aberth method that guarantee that iterations do not terminate until the relative error in the polynomial evaluation is too large or until the relative size of the Ehrlich-Aberth correction term is too small. Moreover, in Corollary 4.3, we showed that under suitable conditions, all root approximations have a limiting accuracy as if computed in twice the working precision and then rounded to the working precision. Finally, in Section 5, extensive numerical experiments illustrate the accuracy of the compensated Horner and Ehrlich-Aberth methods as well as the speed-up in terms of computation time as compared to the quadruple precision implementations of the Horner and Ehrlich-Aberth methods, respectively.

Acknowledgments. The authors are grateful to the referees whose thoughtful comments significantly improved this article.

REFERENCES

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

[2] ANSI/IEEE, *IEEE Standard for Binary Floating Point Arithmetic*, IEEE, New York, std 754-2019 ed., 2019.

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

[4] W. BÖRSCH-SUPAN, *A posteriori error bounds for the zeros of polynomials*, Numer. Math., 5 (1963), pp. 380–398.

[5] F. CHAITIN-CHATELIN AND V. FRAYSSÉ, *Lectures on Finite Precision Computations*, SIAM, Philadelphia, 1996.

[6] T. J. DEKKER, *A floating-point technique for extending the available precision*, Numer. Math., 18 (1971/72), pp. 224–242.

[7] J. DEMMEL AND Y. HIDA, *Accurate and efficient floating point summation*, SIAM J. Sci. Comput., 25 (2003/04), pp. 1214–1248.

[8] L. W. EHRLICH, *A modified Newton method for polynomials*, Commun. ACM, 10 (1967), pp. 107–108.

[9] A. ENGE, M. GASTINEAU, P. THÉVENY, AND P. ZIMMERMANN, *MPC: A library for multiprecision complex arithmetic with exact rounding*, Software, Inria, 1.2.1 ed., 2021. http://mpc.multiprecision.org

[10] L. FOUSSE, G. HANROT, V. LEFÈVRE, P. PÉLISSIER, AND P. ZIMMERMANN, *MPFR: a multiple-precision binary floating-point library with correct rounding*, ACM Trans. Math. Software, 33 (2007), Art. 13, 15 pages.

[11] L. GEMIGNANI, *Accurate polynomial root-finding methods for symmetric tridiagonal matrix eigenproblems*, Comput. Math. Appl., 72 (2016), pp. 992–1001.

[12] S. GILL, *A process for the step-by-step integration of differential equations in an automatic digital computing machine*, Proc. Cambridge Philos. Soc., 47 (1951), pp. 96–108.

[13] D. GOLDBERG, *What every computer scientist should know about floating-point arithmetic*, ACM Comput. Surveys, 23 (1991), pp. 5–48.

[14] S. GRAILLAT, *Accurate simple zeros of polynomials in floating point arithmetic*, Comput. Math. Appl., 56 (2008), pp. 1114–1120.

[15] S. GRAILLAT, P. LANGLOIS, AND N. LOUVET, *Algorithms for accurate, validated and fast polynomial evaluation*, Japan J. Indust. Appl. Math., 26 (2009), pp. 191–214.

[16] S. GRAILLAT, N. LOUVET, AND P. LANGLOIS, *Compensated Horner scheme*, Tech. Rep. RR2005-04, Université de Perpignan Via Domitia, Perpignan 2005.

[17] S. GRAILLAT AND V. MÉNISSIER-MORAIN, *Accurate summation, dot product and polynomial evaluation in complex floating point arithmetic*, Inform. and Comput., 216 (2012), pp. 57–71.

[18] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[19] H. JIANG, S. GRAILLAT, C. HU, S. LI, X. LIAO, L. CHENG, AND F. SU, *Accurate evaluation of the k-th derivative of a polynomial and its application*, J. Comput. Appl. Math., 243 (2013), pp. 28–47.

[20] H. JIANG, S. LI, L. CHENG, AND F. SU, *Accurate evaluation of a polynomial and its derivative in Bernstein form*, Comput. Math. Appl., 60 (2010), pp. 744–755.

[21] W. KAHAN, *Further remarks on reducing truncation errors*, Commun. ACM, 8 (1965), p. 40.

[22] D. E. KNUTH, *The Art of Computer Programming. Vol. 2. Seminumerical Algorithms*, Addison-Wesley, Reading, 1998.

[23] O. MØLLER, *Quasi double-precision in floating point addition*, BIT, 5 (1965), pp. 37–50.

[24] Y. NIEVERGELT, *Scalar fused multiply-add instructions produce floating-point matrix arithmetic provably accurate to the penultimate digit*, ACM Trans. Math. Software, 29 (2003), pp. 27–48.

[25] T. OGITA, S. M. RUMP, AND S. OISHI, *Accurate sum and dot product*, SIAM J. Sci. Comput., 26 (2005), pp. 1955–1988.

[26] D. M. PRIEST, *On Properties of Floating Point Arithmetics: Numerical Stability and the Cost of Accurate Computations*, PhD. Thesis, Computer Science, University of California, Berkeley, 1992.

[27] S. M. RUMP, *Ultimately fast accurate summation*, SIAM J. Sci. Comput., 31 (2009), pp. 3466–3502.

[28] S. M. RUMP, T. OGITA, AND S. OISHI, *Accurate floating-point summation part I: Faithful rounding*, SIAM J. Sci. Comput., 31 (2008), pp. 189–224.

[29] F. TISSEUR, *Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1038–1057.

[30] J. H. WILKINSON, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, 1963.