



HAL
open science

Accurate Horner Methods in Real and Complex Floating-Point Arithmetic

Thomas R Cameron, Stef Graillat

► **To cite this version:**

Thomas R Cameron, Stef Graillat. Accurate Horner Methods in Real and Complex Floating-Point Arithmetic. 2023. hal-04030542

HAL Id: hal-04030542

<https://hal.science/hal-04030542v1>

Preprint submitted on 15 Mar 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Accurate Horner Methods in Real and Complex Floating-Point Arithmetic

Thomas R. Cameron · Stef Graillat

Received: date / Accepted: date

Abstract In this article, we derive accurate Horner methods in real and complex floating-point arithmetic. In particular, we show that these methods are as accurate as if computed in k -fold precision, where $k \geq 2$, and then rounded into the working precision. When $k = 2$, our methods are comparable or faster than the existing compensated Horner routines. When compared to multi-precision software, such as MPFR and MPC, our methods are significantly faster for $k \leq 8$, that is, up to 489 bits in the significand.

Keywords accurate polynomial evaluation, error-free transformations, error analysis, k -fold accuracy

Mathematics Subject Classification (2010) 65Y20, 65-04, 65G99

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 [4, 7, 8, 12, 13, 15]. 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 [5, 6]. More recently, Rump, Ogita, and Oishi have developed algorithms for the summation (SumK) and dot product (DotK) as accurate as if computed in k -fold precision and then rounded into the working precision [17]. Throughout this article, whenever k -fold precision is referenced, it is implicitly assumed that $k \geq 2$, unless otherwise stated.

Even more recently, Rump developed algorithms for the summation (SumKK) and dot product (DotKK) as accurate as if computed in k -fold precision and stored in k parts, which he then used to develop methods for inverting arbitrary ill-conditioned matrices [18]. While Rump only focused on real arithmetic, it is straightforward to extend these methods to complex arithmetic. In this article, we use a similar strategy to develop Horner methods in real and complex arithmetic as accurate as if computed in k -fold precision and then rounded into the working precision. Specifically, we compute each iteration of Horner's method as accurate as if computed in k -fold precision and stored in k parts. After the final iteration, we return the SumK value of the k parts.

The outline of this article is as follows: In Section 2 we recall the basic properties of real and complex floating-point arithmetic, error-free transformations, and the SumK method from [17]. Then, in Section 3, we derive the accurate real and complex Horner methods, which we denote HornerK and HornerKComplex, respectively. Moreover, we prove forward error bounds on these methods that imply each method is

This work was partly supported by the NuSCAP (ANR-20-CE48-0014) project of the French National Agency for Research (ANR).

T. R. Cameron
Mathematics Department
Penn State Behrend
Erie, PA, USA
E-mail: trc5475@psu.edu

S. Graillat
Sorbonne Université
CNRS, LIP6
F-75005 Paris, France
E-mail: stef.graillat@sorbonne-universite.fr

as accurate as if computed in k -fold precision and then rounded into the working precision. Finally, in Section 4, we present the results of several numerical experiments to demonstrate the relative forward error bound and computational time of the HornerK and HornerKCmplx methods.

Note that, when $k = 2$, the HornerK method is comparable to the CompensatedHorner method from [9]. Also, the HornerKCmplx method is comparable to the CompHorner method from [3], though we show that the HornerKCmplx method is significantly faster, especially for large degree polynomials. Finally, we are not the first to develop a Horner's method as accurate as if computed in k -fold precision and then rounded into the working precision, see [14]. However, their method requires $2^k - 1$ evaluations of the EFTHorner method from [9]. For this reason, our method is significantly faster, especially for large values of k .

2 Floating-Point Arithmetic

Throughout this article, we assume that the computer arithmetic satisfies the IEEE 754 standard [2], and that no 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 $\text{fl}(\cdot)$ 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 $\text{fl}(a \circ b)$ to be *correctly rounded*, that is, as accurate as if computed exactly and then rounded to the working precision [8]. In this article, we assume that all the computations are performed with rounding to nearest, using round to even in the case of a tie. As a result, for $a, b \in \mathcal{F}$, floating-point operations satisfy

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

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

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

Throughout this article, we make use of the quantity:

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

where $n \in \mathbb{N}$ is assumed to satisfy $n\mu < 1$. In addition, for $\mathbf{x}, \mathbf{y} \in \mathcal{F}^n$, we make use of the following error bound on the floating-point summation:

$$\left| \text{fl} \left(\sum_{i=1}^n x_i \right) - \sum_{i=1}^n x_i \right| \leq \gamma_{n-1} \sum_{i=1}^n |x_i|$$

and the following error bound on the floating-point dot-product:

$$|\text{fl}(\mathbf{x} \cdot \mathbf{y}) - \mathbf{x} \cdot \mathbf{y}| \leq \gamma_n |\mathbf{x}| \cdot |\mathbf{y}|.$$

Both bounds are proven in [11] and from their proofs it is clear that similar bounds hold for the complex floating-point summation and the complex floating-point dot-product.

For each $x = \text{fl}(a \circ b)$, 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 [13] and returns the error-free transformation of (a, b) for addition.

Algorithm 1 Error-free transformation of $(a, b) \in \mathcal{F}^2$ for addition [13, Thm B, p.236].

```
function [x, y] = TwoSum(a, b) :
    x = fl(a + b)
    z = fl(x - a)
    y = fl((a - (x - z)) + (b - z))
```

The fused multiply-add operation, denoted FMA (a, b, c) , results in the floating-point number nearest to $a \cdot b + c \in \mathbb{R}$. We make use of the FMA operation to perform the error-free transformation of (a, b) for multiplication, see Algorithm 2. Note that the fused multiply-add operation was added to the IEEE 754 standard in 2008 and since 2003 has been supported by many modern processors [16].

Algorithm 2 Error-free transformation of $(a, b) \in \mathcal{F}^2$ for multiplication [16, Thm 2].

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

Theorem 1 summarizes the properties of Algorithm 1 and Algorithm 2. Note that we are able to reduce the 17 flops for the TwoProd function, as stated in [17, Theorem 3.4], to 2 flops since we use the FMA operation rather than the Split function from [4].

Theorem 1 ([17, Thm 3.4]) *Let $a, b \in \mathcal{F}$. Then, $[x, y] = \text{TwoSum}(a, b)$ requires 6 flops and satisfies*

$$a + b = x + y, \quad x = \text{fl}(a + b), \quad |y| \leq \mu |x|, \quad |y| \leq \mu |a + b|.$$

Also, $[x, y] = \text{TwoProd}(a, b)$ requires 2 flops and satisfies

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

Next, we state the vector transformation from [17], also known as the distillation algorithm [13], see Algorithm 3, where we do not overwrite the input vector for clarity in the analysis of the algorithm. Theorem 2 summarizes several important properties of Algorithm 3. Note that a similar result holds in complex floating-point arithmetic, see Theorem 6.

Algorithm 3 Transformation of the vector $\mathbf{p} \in \mathcal{F}^n$ without changing the vector sum of \mathbf{p} ([17, Alg 4.3])

```
function  $\mathbf{q} = \text{VecSum}(\mathbf{p})$  :
   $q_1 = p_1$ 
  for  $i = 2, \dots, n$  do
     $[q_i, q_{i-1}] = \text{TwoSum}(p_i, q_{i-1})$ 
  end for
```

Theorem 2 ([17, Lemma 4.2]) *Let $\mathbf{p} \in \mathcal{F}^n$. Then, $\mathbf{q} = \text{VecSum}(\mathbf{p})$ requires $6(n-1)$ flops and satisfies*

$$\sum_{i=1}^n q_i = \sum_{i=1}^n p_i,$$

$q_n = \text{fl}(\sum_{i=1}^n p_i)$, and

$$\sum_{i=1}^{n-1} |q_i| \leq \gamma_{n-1} \sum_{i=1}^n |p_i|.$$

We conclude this section with the k -fold summation from [17], see Algorithm 4. Theorem 3 summarizes the properties of Algorithm 4. Note that $3\gamma_{n-1}^2$ is negligible compared with μ . So, the result of Theorem 3 tells us that the result $s = \text{SumK}(\mathbf{p}, k)$ is as accurate as if computed in k -fold precision and then rounded into the working precision.

Algorithm 4 Vector Summation of $\mathbf{p} \in \mathcal{F}^n$ in k -fold precision and rounded into the working precision [17, Alg 4.8]

```
function  $s = \text{SumK}(\mathbf{p}, k)$  :
  for  $i = 1, \dots, k-1$  do
     $\mathbf{p} = \text{VecSum}(\mathbf{p})$ 
  end for
   $s = \text{fl}(\left(\sum_{i=1}^{n-1} p_i\right) + p_n)$ 
```

Theorem 3 ([17, Prop 4.5 and 4.10]) Let $\mathbf{p} \in \mathcal{F}^n$, $4n\mu \leq 1$, and $k \geq 2$. Then, $s = \text{SumK}(\mathbf{p}, k)$ requires $(n-1)(6k-5)$ flops and satisfies

$$\left| s - \sum_{i=1}^n p_i \right| \leq (\mu + 3\gamma_{n-1}^2) \left| \sum_{i=1}^n p_i \right| + \gamma_{2n-2}^k \sum_{i=1}^n |p_i|.$$

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 $\text{Re}(\cdot)$ and $\text{Im}(\cdot)$ to denote the real and imaginary part of a complex number, respectively. As in the real case, we denote by $\text{fl}(\cdot)$ the operations that are done in floating-point working precision. The following holds for all $a, b \in \mathcal{C}$ and $\circ \in \{+, -\}$:

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

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

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

where $|\epsilon| \leq \sqrt{2}\gamma_2$. This further implies that for $\circ \in \{+, -\}$, we have

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

and

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

Throughout this article, we make use of the quantity

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

where $n \in \mathbb{N}$ is assumed to satisfy $n\sqrt{2}\gamma_2 < 1$. In addition, for $\mathbf{x}, \mathbf{y} \in \mathcal{C}^n$, we make use of the following error bound on the complex floating-point summation:

$$\left| \text{fl} \left(\sum_{i=1}^n x_i \right) - \sum_{i=1}^n x_i \right| \leq \gamma_{n-1} \sum_{i=1}^n |x_i|$$

and the following error bound on the floating-point dot-product:

$$|\text{fl}(\mathbf{x} \cdot \mathbf{y}) - \mathbf{x} \cdot \mathbf{y}| \leq \tilde{\gamma}_n |\mathbf{x}| \cdot |\mathbf{y}|.$$

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 $x = \text{fl}(a \circ b)$ and $x + y = a \circ b$. The error-free transformation of $(a, b) \in \mathcal{C}^2$ for complex addition is a straightforward extension of Algorithm 1 and is shown in Algorithm 5. In contrast, the error-free transformation of $(a, b) \in \mathcal{C}^2$ for complex multiplication requires multiple

Algorithm 5 Error-free transformation of $(a, b) \in \mathcal{C}^2$ for addition ([10, Alg 3.1]).

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

products of the real and imaginary parts of a and b as shown in Algorithm 6. Note that Theorem 4

Algorithm 6 Error-free transformation of $(a, b) \in \mathcal{C}^2$ for multiplication ([10, Alg 3.2]).

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

summarizes the properties of Algorithm 5 and Algorithm 6.

Theorem 4 ([10, Thm 3.1 and 3.2]) Let $a, b \in \mathcal{C}$. Then, $[x, y] = \text{TwoSumCmplx}(a, b)$ requires 12 flops and satisfies

$$a + b = x + y, \quad x = \text{fl}(a + b), \quad |y| \leq \mu |x|, \quad |y| \leq \mu |a + b|.$$

Also, $[w, x, y, z] = \text{TwoProdCmplx}(a, b)$ requires 20 flops and satisfies

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

In addition, we have the following result for Algorithm 6.

Theorem 5 Let $a, b \in \mathcal{C}$ and let $[w, x, y, z] = \text{TwoProdCmplx}(a, b)$. Then,

$$|x| + |y| + |z| \leq \mu \left(3 + \sqrt{2}\gamma_2\right) |a| |b|.$$

Proof Note that $x = h_1 + ih_3$, where $[g_1, h_1] = \text{TwoProd}(\text{Re}(a), \text{Re}(b))$ and $[g_3, h_3] = \text{TwoProd}(\text{Re}(a), \text{Im}(b))$. Therefore, by Theorem 1, we have

$$\begin{aligned} |x| &= \sqrt{h_1^2 + h_3^2} \\ &\leq \sqrt{\mu^2 \text{Re}(a)^2 \text{Re}(b)^2 + \mu^2 \text{Re}(a)^2 \text{Im}(b)^2} \\ &= \mu |\text{Re}(a)| |b| \leq \mu |a| |b|. \end{aligned}$$

A similar argument shows that $|y| \leq \mu |a| |b|$. Finally, for $z = h_5 + ih_6$, where $[g_5, h_5] = \text{TwoSum}(g_1, -g_2)$ and $[g_6, h_6] = \text{TwoSum}(g_3, g_4)$. Theorem 1 implies that $|h_5| \leq \mu |g_5|$ and $|h_6| \leq \mu |g_6|$. Therefore, we have

$$\begin{aligned} |z| &= \sqrt{h_5^2 + h_6^2} \\ &\leq \sqrt{\mu^2 g_5^2 + \mu^2 g_6^2} \\ &= \mu \sqrt{g_5^2 + g_6^2} \\ &= \mu |w| \\ &= \mu |\text{fl}(a \cdot b)| \\ &\leq \mu \left(1 + \sqrt{2}\gamma_2\right) |a| |b|. \end{aligned}$$

Next, we state the complex vector transformation, see Algorithm 7 where we do not overwrite the input vector for clarity in the analysis of the algorithm. Theorem 6 summarizes several important properties of Algorithm 7. Note that the proof of Theorem 6 follows from Theorem 2 since the properties of TwoSum and TwoSumCmplx are identical.

Algorithm 7 Transformation of the vector $\mathbf{p} \in \mathcal{C}^n$ without changing the vector sum of \mathbf{p} .

```
function  $\mathbf{q} = \text{VecSumCmplx}(\mathbf{p})$  :
   $q_1 = p_1$ 
  for  $i = 2, \dots, n$  do
     $[q_i, q_{i-1}] = \text{TwoSumCmplx}(p_i, q_{i-1})$ 
  end for
```

Theorem 6 Let $\mathbf{p} \in \mathcal{C}^n$. Then, $\mathbf{q} = \text{VecSumCmplx}(\mathbf{p})$ requires $12(n - 1)$ flops and satisfies

$$\sum_{i=1}^n q_i = \sum_{i=1}^n p_i,$$

$q_n = \text{fl}\left(\sum_{i=1}^n p_i\right)$, and

$$\sum_{i=1}^{n-1} |q_i| \leq \gamma_{n-1} \sum_{i=1}^n |p_i|.$$

Algorithm 8 Vector Summation of $\mathbf{p} \in \mathcal{C}^n$ in k -fold precision and rounded into the working precision.

```
function  $s = \text{SumKCmplx}(\mathbf{p}, k)$  :
  for  $i = 1, \dots, k - 1$  do
     $\mathbf{p} = \text{VecSumCmplx}(\mathbf{p})$ 
  end for
   $s = \text{fl}\left(\left(\sum_{i=1}^{n-1} p_i\right) + p_n\right)$ 
```

We conclude this section with the complex k -fold summation, see Algorithm 8. Theorem 7 summarizes the properties of Algorithm 8. Note that the proof of Theorem 7 follows from Theorem 3 since the properties of `VecSumCmplx` and `VecSum` are identical. Also, the $3\gamma_{n-1}^2$ is negligible compared with μ . So, the result of Theorem 7 tells us that the result $s = \text{SumKCmplx}(\mathbf{p}, k)$ is as accurate as if computed in k -fold precision and then rounded into the working precision.

Theorem 7 Let $\mathbf{p} \in \mathcal{C}^n$, $4n\mu \leq 1$, and $k \geq 2$. Then, $s = \text{SumKCmplx}(\mathbf{p}, k)$ requires $2(n-1)(6k-5)$ flops and satisfies

$$\left|s - \sum_{i=1}^n p_i\right| \leq (\mu + 3\gamma_{n-1}^2) \left|\sum_{i=1}^n p_i\right| + \gamma_{2n-2}^k \sum_{i=1}^n |p_i|.$$

3 Horner's Method

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

$$p(z) = a_m z^m + \dots + a_1 z + a_0, \quad (1)$$

where a_0, a_1, \dots, a_m are real or complex floating-point numbers, and $a_m \neq 0$. If all coefficients are real, then we write $p \in \mathcal{F}[z]$, and if any of the coefficients are complex, then we write $p \in \mathcal{C}[z]$. Given a real or complex floating-point number z , Algorithm 9 defines Horner's method, which is used to compute the polynomial evaluation $p(z)$. For $i = 0, 1, \dots, m$, we use h_i to denote the i th step of Horner's method computed in exact arithmetic, and we use H_i to denote the value obtained when all operands are replaced by their respective absolute value. For example, $h_0 = p(z)$ and $H_0 = \tilde{p}(|z|)$, where $\tilde{p}(z) = \sum_{i=0}^m |a_i| z^i$.

Algorithm 9 Horner's method in exact arithmetic [11, p.94].

```
function  $h_0 = \text{Horner}(p, z)$  :
   $h_m = a_m$ 
  for  $i = m - 1$  to  $i = 0$  do
     $h_i = z \cdot h_{i+1} + a_i$ 
  end for
```

If real or complex floating-point arithmetic in the working precision is used, then we denote the result of each step of Horner's method by $\text{fl}(h_i)$, for $i = 0, 1, \dots, m$. It is well-known that in real floating-point arithmetic, we have the following error bound for h_0 [11]:

$$|h_0 - \text{fl}(h_0)| \leq \gamma_{2m} \tilde{p}(|z|). \quad (2)$$

Similarly, in complex floating-point arithmetic, we have [10]:

$$|h_0 - \text{fl}(h_0)| \leq \tilde{\gamma}_{2m} \tilde{p}(|z|). \quad (3)$$

In what follows, we derive methods for the real and complex Horner's method as accurate as if computed in k -fold precision and then rounded into the working precision.

3.1 Real HornerK Method

The real Horner's method in k -fold precision and then rounded into the working precision is shown in Algorithm 10. Note that $\{h^{(i)}\}$ is a list of floating-point numbers that denote the k -fold result of Horner's method on the i -th iteration. Starting with $\{h^{(m)}\} = \{a_m, 0, \dots, 0\}$, we let $\{h^{(i)}\}$ denote the k -fold result of Horner's method on the i -th iteration stored in k parts, for $i = m-1, \dots, 0$. In particular, $h_1^{(i)}$ is the

Algorithm 10 Horner's method in k -fold precision and rounded into the working precision.

```

1: function  $h = \text{HornerK}(p, z, k)$  :
2:  $\{h^{(m)}\} = \{a_m, 0, \dots, 0\}$ , vector of size  $k$ 
3: for  $i = m - 1$  to  $i = 0$  do
4:    $[r, e_1] = \text{TwoProd}(z, h_1^{(i+1)})$ 
5:   for  $j = 2$  to  $j = k$  do
6:      $[s, e_j] = \text{TwoProd}(z, h_j^{(i+1)})$ 
7:      $[r, e_{k+j-1}] = \text{TwoSum}(r, s)$ 
8:   end for
9:    $[h_1^{(i)}, e_{2k}] = \text{TwoSum}(r, a_i)$ 
10:  Set  $\mathbf{e}^{(0)} = \mathbf{e}$ 
11:  for  $j = 0$  to  $j = k - 3$  do
12:     $\mathbf{e}^{(j+1)} = \text{VecSum}(\mathbf{e}^{(j)})$ 
13:     $h_{j+2}^{(i)} = e_{2k-j}^{(j+1)}$ 
14:    Delete entry  $e_{2k-j}^{(j+1)}$  from  $\mathbf{e}^{(j+1)}$ 
15:  end for
16:   $h_k^{(i)} = \text{fl}\left(\sum_{j=1}^{k+2} e_j^{(k-2)}\right)$ 
17: end for
18:  $h = \text{SumK}(\{h^{(0)}\}, k)$ 

```

floating-point result of Horner's method on the i -th iteration and $\mathbf{e}^{(0)}$ is a vector of size $2k$ that stores the errors in that computation. Then, for $j = 0, \dots, k - 3$, we set $h_{j+2}^{(i)}$ to the floating-point sum of the entries in $\mathbf{e}^{(j)}$, and we set $\mathbf{e}^{(j+1)}$ to be a vector of size $2k - (j + 1)$ that stores the errors in that computation.

In what follows, we prove bounds on the forward error of the floating-point result from Algorithm 10. To begin, Lemma 1 gives a bound on the sum of the absolute value of the entries of \mathbf{e} .

Lemma 1 *The vector \mathbf{e} in Algorithm 10 satisfies*

$$\sum_{j=1}^{2k} |e_j| \leq \gamma_{k+1} \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right),$$

for $i = m - 1, \dots, 0$.

Proof By Theorem 1, we have

$$\sum_{j=1}^k |e_j| \leq \mu \sum_{j=1}^k |z \cdot h_j^{(i+1)}|.$$

Also, by Theorem 2, we have

$$\begin{aligned} \sum_{j=k+1}^{2k} |e_j| &\leq \gamma_k \left(\sum_{j=1}^k \left| \text{fl}(z \cdot h_j^{(i+1)}) \right| + |a_i| \right) \\ &\leq \gamma_k \left((1 + \mu) \sum_{j=1}^k |z \cdot h_j^{(i+1)}| + |a_i| \right) \\ &\leq \gamma_k (1 + \mu) \left(\sum_{j=1}^k |z \cdot h_j^{(i+1)}| + |a_i| \right). \end{aligned}$$

Since $\mu + (1 + \mu)\gamma_k \leq \gamma_{k+1}$, we have

$$\sum_{j=1}^{2k} |e_j| \leq \gamma_{k+1} \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right).$$

Next, Lemma 2 shows that the collection $\{h^{(i)}\}$ stores the floating-point result of $z \sum_{j=1}^k h_j^{(i+1)} + a_i$ computed as if in k -fold precision and stored in k -parts.

Lemma 2 *The floating-point collection $\{h^{(i)}\}$ in Algorithm 10 satisfies*

$$\left| \sum_{j=1}^k h_j^{(i)} - \left(z \sum_{j=1}^k h_j^{(i+1)} + a_i \right) \right| \leq \gamma_{2^{k-1}}^k \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right)$$

and

$$\sum_{j=1}^k |h_j^{(i)}| \leq (1 + 2\gamma_{2^{k-1}} + \dots + 2\gamma_{2^{k-1}}^{k-1} + \gamma_{2^{k-1}}^k) \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right),$$

for $i = m-1, \dots, 0$.

Proof By successive application of Theorem 1, we have from line 4 – 9 of Algorithm 10 that

$$h_1^{(i)} + \sum_{j=1}^{2k} e_j = z \sum_{j=1}^k h_j^{(i+1)} + a_i.$$

Furthermore, by successive application of Theorem 2, we have from line 10 – 15 of Algorithm 10 that

$$\sum_{j=1}^{2k} e_j = \sum_{j=2}^{k-1} h_j^{(i)} + \sum_{j=1}^{k+2} e_j^{(k-2)},$$

where

$$\sum_{j=1}^{k+2} |e_j^{(k-2)}| \leq \gamma_{k+2} \sum_{j=1}^{k+3} |e_j^{(k-3)}| \leq \dots \leq \left(\prod_{j=k+2}^{2k-1} \gamma_j \right) \sum_{j=1}^{2k} |e_j| \leq \gamma_{2^{k-1}}^{k-2} \sum_{j=1}^{2k} |e_j|.$$

Also, in line 16 of Algorithm 10, the error in the floating-point summation satisfies

$$\left| h_k^{(i)} - \sum_{j=1}^{k+2} e_j^{(k-2)} \right| \leq \gamma_{k+1} \sum_{j=1}^{k+2} |e_j^{(k-2)}|.$$

Combining these observations with Lemma 1 gives

$$\begin{aligned} \left| \sum_{j=1}^k h_j^{(i)} - \left(z \sum_{j=1}^k h_j^{(i+1)} + a_i \right) \right| &= \left| h_k^{(i)} - \sum_{j=1}^{k+2} e_j^{(k-2)} \right| \\ &\leq \gamma_{k+1} \sum_{j=1}^{k+2} |e_j^{(k-2)}| \\ &\leq \gamma_{k+1} \gamma_{2^{k-1}}^{k-2} \sum_{j=1}^{2k} |e_j| \\ &\leq \gamma_{k+1} \gamma_{2^{k-1}}^{k-2} \gamma_{k+1} \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right) \\ &\leq \gamma_{2^{k-1}}^k \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right). \end{aligned}$$

Note that Theorem 1 implies that $h_1^{(i)} = \text{fl} \left(z \sum_{j=1}^k h_j^{(i+1)} + a_i \right)$ and Theorem 2 implies that

$$h_l^{(i)} = \text{fl} \left(\sum_{j=1}^{2k-l+2} e_j^{(l-2)} \right),$$

for $l = 2, \dots, k$. Therefore, the error in floating-point dot product gives us the following bound

$$\begin{aligned} |h_1^{(i)}| &\leq (1 + \gamma_{k+1}) \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right) \\ &\leq (1 + \gamma_{2k-1}) \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right). \end{aligned}$$

Similarly, the error in floating-point summation gives us the following bound

$$\begin{aligned} |h_l^{(i)}| &\leq (1 + \gamma_{2k-l+1}) \sum_{j=1}^{2k-l+2} |e_j^{(l-2)}| \\ &\leq (1 + \gamma_{2k-l+1}) \gamma_{2k-l+2} \cdots \gamma_{2k-1} \sum_{j=1}^{2k} |e_j| \\ &\leq (\gamma_{2k-1}^{l-1} + \gamma_{2k-1}^l) \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right), \end{aligned}$$

where the last line follows from Lemma 1.

Next, Theorem 8 provides an error bound on the sum of the entries in the collection $\{h^{(i)}\}$ versus the exact value of Horner's method on the i -th iteration. To that end, let $\rho = 1 + 2\gamma_{2k-1} + \cdots + 2\gamma_{2k-1}^{k-1} + \gamma_{2k-1}^k$. Also, recall that for $i = 0, 1, \dots, m$, h_i denotes the exact value from Algorithm 9 and H_i denotes the exact value when all operands are replaced by their respective absolute value.

Theorem 8 *The floating-point collection $\{h^{(m-i)}\}$ in Algorithm 10 satisfies*

$$\left| \sum_{j=1}^k h_j^{(m-i)} - h_{m-i} \right| \leq \gamma_{2k-1}^k (1 + \rho + \cdots + \rho^{i-1}) H_{m-i}$$

and

$$\sum_{j=1}^k |h_j^{(m-i)}| \leq \rho^i H_{m-i},$$

for $i = 0, \dots, m$.

Proof We proceed via induction on i . The base case, when $i = 0$, is clear. Now, suppose the result holds for some $i \geq 0$, and note that the triangle inequality implies

$$\left| \sum_{j=1}^k h_j^{(m-i-1)} - h_{m-i-1} \right| \leq \left| \sum_{j=1}^k h_j^{(m-i-1)} - \left(z \sum_{j=1}^k h_j^{(m-i)} + a_{m-i-1} \right) \right| + |z| \left| \sum_{j=1}^k h_j^{(m-i)} - h_{m-i} \right|.$$

Applying Lemma 2 and the induction hypothesis gives us

$$\begin{aligned} \left| \sum_{j=1}^k h_j^{(m-i-1)} - h_{m-i-1} \right| &\leq \gamma_{2k-1}^k \left(|z| \sum_{j=1}^k |h_j^{(m-i)}| + |a_{m-i-1}| \right) + |z| \gamma_{2k-1}^k (1 + \rho + \cdots + \rho^{i-1}) H_{m-i} \\ &\leq \gamma_{2k-1}^k (|z| \rho^i H_{m-i} + |a_{m-i-1}|) + |z| \gamma_{2k-1}^k (1 + \rho + \cdots + \rho^{i-1}) H_{m-i} \\ &\leq \gamma_{2k-1}^k (1 + \rho + \cdots + \rho^i) H_{m-i-1}. \end{aligned}$$

Moreover, by Lemma 2, we have

$$\begin{aligned} \sum_{j=1}^k |h_j^{(m-i-1)}| &\leq \rho \left(|z| \sum_{j=1}^k |h_j^{(m-i)}| + |a_{m-i-1}| \right) \\ &\leq \rho (|z| \rho^i H_{m-i} + |a_{m-i-1}|) \leq \rho^{i+1} H_{m-i-1}, \end{aligned}$$

and so the result holds for $i + 1$.

Finally, Corollary 1 gives a forward error bound and the flop count for Algorithm 10. Note that when $k = 2$, the flop count is $19m + 7$, which is slightly greater than the $11m - 1$ flops for the CompensatedHorner method from [9].

Corollary 1 *Let $p \in \mathcal{F}[z]$ be a degree m polynomial, $z \in \mathcal{F}$, $4n\mu \leq 1$, and $k \geq 2$. Then, $h = \text{HornerK}(p, z, k)$ requires $m(9k^2 - 6k - 5) + (6k^2 - 11k + 5)$ flops and satisfies*

$$|h - p(z)| \leq (\mu + 3\gamma_{k-1}^2) |p(z)| + 2\gamma_{2k-1}^k (1 + \rho + \dots + \rho^m) \tilde{p}(|z|).$$

Proof The flop count is left to the reader. For the error bound, note that the triangle inequality implies

$$|h - p(z)| \leq \left| h - \sum_{j=1}^k h_j^{(0)} \right| + \left| \sum_{j=1}^k h_j^{(0)} - h_0 \right|.$$

Applying Theorem 3 and Theorem 8 gives us

$$\begin{aligned} |h - p(z)| &\leq (\mu + 3\gamma_{k-1}^2) \left| \sum_{j=1}^k h_j^{(0)} \right| + \gamma_{2k-2}^k \sum_{j=1}^k |h_j^{(0)}| + \gamma_{2k-1}^k (1 + \rho + \dots + \rho^{m-1}) H_0 \\ &\leq (\mu + 3\gamma_{k-1}^2) (|h_0| + \gamma_{2k-1}^k (1 + \rho + \dots + \rho^{m-1}) H_0) + \gamma_{2k-1}^k (1 + \rho + \dots + \rho^m) H_0 \\ &\leq (\mu + 3\gamma_{k-1}^2) |p(z)| + 2\gamma_{2k-1}^k (1 + \rho + \dots + \rho^m) \tilde{p}(|z|). \end{aligned}$$

It may be unsettling to see the error bound in Corollary 1 have the term $(1 + \rho + \dots + \rho^m)$; however, the following proposition shows that this term can be replaced by $(m + 4)$ for reasonably sized k and m

Proposition 1 *For $2 \leq k \leq 10$ and $1 \leq m \leq 10^5$, we have*

$$1 + \rho + \dots + \rho^m \leq (m + 4).$$

Proof Note that

$$1 + \rho + \dots + \rho^m = \frac{\rho^{m+1} - 1}{\rho - 1},$$

where $\rho = 1 + 2\gamma_{2k-1} + \dots + 2\gamma_{2k-1}^{k-1} + \gamma_{2k-1}^k$. Define $\theta = 2\gamma_{2k-1} + \dots + 2\gamma_{2k-1}^{k-1} + \gamma_{2k-1}^k$, then $\rho = 1 + \theta$ and $\rho^{m+1} = \sum_{j=0}^{m+1} \binom{m+1}{j} \theta^j$. Now, it is true that $\binom{m+1}{j} \theta^j > \binom{m+1}{j+1} \theta^{j+1}$, for $j = 0, 1, \dots, m$; otherwise, there is a j for which $\theta \geq \frac{j+1}{m+1}$, which contradicts the reasonably sized k and m . Therefore, we have

$$\begin{aligned} \rho^{m+1} &= 1 + (m+1)\theta + \sum_{j=2}^{m+1} \binom{m+1}{j} \theta^j \\ &\leq 1 + (m+1)\theta + \sum_{j=2}^{m+1} \frac{(m+1)!}{2(m-1)!} \theta^2 \\ &= 1 + (m+1)\theta + \frac{m^2(m+1)}{2} \theta^2 \\ &\leq 1 + (m+4)\theta, \end{aligned}$$

where the last line follows from $m^2(m+1)\theta^2 \leq 6\theta$; otherwise, $\theta > \frac{6}{m^2(m+1)}$, which contradicts the reasonably sized k and m . Finally, we have

$$1 + \rho + \dots + \rho^m = \frac{\rho^{m+1} - 1}{\rho - 1} \leq \frac{(m+4)\theta}{\theta} = (m+4).$$

Now, we can re-write the error bound from Corollary 1. Note that $3\gamma_{k-1}^2$ is negligible compared with μ . So, the result of Corollary 2 tells us that the result $h = \text{HornerK}(\mathbf{p}, z, k)$ is as accurate as if computed in k -fold precision and then rounded into the working precision.

Corollary 2 *Let $p \in \mathcal{F}[z]$ be a degree m polynomial, where $1 \leq m \leq 10^5$. Also, let $z \in \mathcal{F}$ and $2 \leq k \leq 10$. Then, $h = \text{HornerK}(p, z, k)$ satisfies*

$$|h - p(z)| \leq (\mu + 3\gamma_{k-1}^2) |p(z)| + 2(m+4)\gamma_{2k-1}^k \tilde{p}(|z|).$$

3.2 Complex HornerK Method

The complex Horner's method in k -fold precision and then rounded into the working precision is shown in Algorithm 11. As in the real case, the vector of floating-point numbers $\{h^{(i)}\}$ denotes the k -fold result of Horner's method on the i -th iteration. In particular, $h_1^{(i)}$ is the floating-point result of the i -th iteration of Horner's method and $\mathbf{e}^{(0)}$ is a vector of size $4k$ that stores the errors in that computation. Then, for $j = 0, \dots, k-3$, we set $h_{j+2}^{(i)}$ to the floating-point sum of the entries in $\mathbf{e}^{(j)}$, and we set $\mathbf{e}^{(j+1)}$ to the vector of size $4k - (j+1)$ that stores the error in that computation.

Algorithm 11 Complex Horner's method in k -fold precision and rounded into the working precision.

```

1: function  $h = \text{HornerKCmplx}(p, z, k)$  :
2:  $\{h^{(m)}\} = \{a_m, 0, \dots, 0\}$ , vector of size  $k$ 
3: for  $i = m - 1$  to  $i = 0$  do
4:    $[r, e_1, e_2, e_3] = \text{TwoProdCmplx}(z, h_1^{(i+1)})$ 
5:   for  $j = 2$  to  $j = k$  do
6:      $[s, e_{3j-2}, e_{3j-1}, e_{3j}] = \text{TwoProdCmplx}(z, h_j^{(i+1)})$ 
7:      $[r, e_{3k+j-1}] = \text{TwoSumCmplx}(r, s)$ 
8:   end for
9:    $[h_1^{(i)}, e_{4k}] = \text{TwoSumCmplx}(r, a_i)$ 
10:  Set  $\mathbf{e}^{(0)} = \mathbf{e}$ 
11:  for  $j = 0$  to  $j = k - 3$  do
12:     $\mathbf{e}^{(j+1)} = \text{VecSumCmplx}(\mathbf{e}^{(j)})$ 
13:     $h_{j+2}^{(i)} = e_{4k-j}^{(j+1)}$ 
14:    Delete entry  $e_{4k-j}^{(j+1)}$  from  $\mathbf{e}^{(j+1)}$ 
15:  end for
16:   $h_k^{(i)} = \text{fl}\left(\sum_{j=1}^{3k+2} e_j^{(k-2)}\right)$ 
17: end for
18:  $h = \text{SumKCmplx}(\{h^{(0)}\}, k)$ 

```

In what follows, we prove bounds on the forward error of the floating-point result from Algorithm 11. To begin, Lemma 3 gives a bound on the sum of the absolute value of the entries of \mathbf{e} .

Lemma 3 *The vector \mathbf{e} in Algorithm 11 satisfies*

$$\sum_{j=1}^{4k} |e_j| \leq \tilde{\gamma}_{k+2} \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right)$$

Proof By Theorem 5, for $j = 1, \dots, k$, we have

$$|e_{3j-2}| + |e_{3j-1}| + |e_{3j}| \leq (3\mu + \sqrt{2}\gamma_2\mu) |z| |h_j^{(i+1)}|,$$

which implies that

$$\sum_{j=1}^{3k} |e_j| \leq \mu (3 + \sqrt{(2)}\gamma_2) |z| \sum_{j=1}^k |h_j^{(i+1)}|.$$

Also, by Theorem 6, we have

$$\begin{aligned} \sum_{j=3k+1}^{4k} &\leq \gamma_k \left(\sum_{j=1}^k |fl(z \cdot h_j^{(i+1)})| + |a_i| \right) \\ &\leq \gamma_k (1 + \sqrt{2}\gamma_2) \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right). \end{aligned}$$

Since $3\mu + \sqrt{2}\gamma_2\mu + \gamma_k (1 + \sqrt{2}\gamma_2) \leq \tilde{\gamma}_{k+2}$, it follows that

$$\sum_{j=1}^{4k} |e_j| \leq \tilde{\gamma}_{k+2} \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right).$$

Next, Lemma 4 shows that the collection $\{h^{(i)}\}$ stores the floating-point result of $z \sum_{j=1}^k h_j^{(i+1)} + a_i$ computed as if in k -fold precision and stored in k -parts.

Lemma 4 *The floating-point collection $\{h^{(i)}\}$ in Algorithm 11 satisfies*

$$\left| \sum_{j=1}^k h_j^{(i)} - \left(z \sum_{j=1}^k h_j^{(i+1)} + a_i \right) \right| \leq \tilde{\gamma}_{4k-1}^k \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right)$$

and

$$\sum_{j=1}^k |h_j^{(i)}| \leq (1 + 2\tilde{\gamma}_{4k-1} + \cdots + 2\tilde{\gamma}_{4k-1}^{k-1} + \tilde{\gamma}_{4k-1}^k) \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right).$$

Proof By successive application of Theorem 4, we have from lines 4 – 9 of Algorithm 11 that

$$h_1^{(i)} + \sum_{j=1}^{4k} e_j = z \sum_{j=1}^k h_j^{(i+1)} + a_i.$$

Furthermore, by successive application of Theorem 6, we have from lines 10 – 15 of Algorithm 11 that

$$\sum_{j=1}^{4k} e_j = \sum_{j=2}^{k-1} h_j^{(i)} + \sum_{j=1}^{3k+2} e_j^{(k-2)},$$

where

$$\sum_{j=1}^{3k+2} |e_j^{(k-2)}| \leq \gamma_{3k+2} \sum_{j=1}^{3k+3} |e_j^{(k-3)}| \leq \cdots \leq \left(\prod_{j=3k+2}^{4k-1} \gamma_j \right) \sum_{j=1}^{4k} |e_j| \leq \gamma_{4k-1}^{k-2} \sum_{j=1}^{4k} |e_j|.$$

Also, in line 16 of Algorithm 11, the error in the floating-point summation satisfies

$$\left| h_k^{(i)} - \sum_{j=1}^{3k+2} e_j^{(k-2)} \right| \leq \gamma_{3k+1} \sum_{j=1}^{3k+2} |e_j^{(k-2)}|.$$

Combining these observations with Lemma 3 gives

$$\begin{aligned} \left| \sum_{j=1}^k h_j^{(i)} - \left(z \sum_{j=1}^k h_j^{(i+1)} + a_i \right) \right| &= \left| h_k^{(i)} - \sum_{j=1}^{3k+2} e_j^{(k-2)} \right| \\ &\leq \gamma_{3k+1} \sum_{j=1}^{3k+2} |e_j^{(k-2)}| \\ &\leq \gamma_{3k+1} \gamma_{4k-1}^{k-2} \sum_{j=1}^{4k} |e_j| \\ &\leq \gamma_{3k+1} \gamma_{4k-1}^{k-2} \tilde{\gamma}_{k+2} \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right) \\ &\leq \tilde{\gamma}_{4k-1}^k \left(|z| \sum_{j=1}^k |h_j^{(i+1)}| + |a_i| \right). \end{aligned}$$

Note that Theorem 4 implies that $h_1^{(i)} = \text{fl} \left(z \sum_{j=1}^k h_j^{(i+1)} + a_i \right)$ and Theorem 6 implies that

$$h_l^{(i)} = \text{fl} \left(\sum_{j=1}^{4k-l+2} e_j^{(l-2)} \right),$$

for $l = 2, \dots, k$. Therefore, the error in the complex floating-point dot product gives us the following bound

$$\left| h_1^{(i)} \right| \leq (1 + \tilde{\gamma}_{k+1}) \left(|z| \sum_{j=1}^k \left| h_j^{(i+1)} \right| + |a_i| \right).$$

Similarly, for $l = 2, \dots, k$, the error in the complex floating-point summation gives us the following bound

$$\begin{aligned} \left| h_l^{(i)} \right| &\leq (1 + \gamma_{4k-l+1}) \sum_{j=1}^{4k-l+2} \left| e_j^{(l-2)} \right| \\ &\leq (1 + \gamma_{4k-l+1}) \gamma_{4k-l+2} \cdots \gamma_{4k-1} \sum_{j=1}^{4k} |e_j| \\ &\leq (\tilde{\gamma}_{4k-1}^{l-1} + \tilde{\gamma}_{4k-1}^l) \left(|z| \sum_{j=1}^k \left| h_j^{(i+1)} \right| + |a_i| \right), \end{aligned}$$

where the last line follows from Lemma 3.

Next, Theorem 9 provides an error bound on the sum of the entries in the collection $\{h^{(i)}\}$ versus the exact value of Horner's method on the i -th iteration. To that end, let $\tilde{\rho} = 1 + 2\tilde{\gamma}_{4k-1} + \cdots + 2\tilde{\gamma}_{4k-1}^{k-1} + \tilde{\gamma}_{4k-1}^k$. Also, recall that for $i = 0, 1, \dots, m$, h_i denotes the exact value from Algorithm 9 and H_i denotes the exact value when all operands are replaced by their respective absolute value.

Theorem 9 *The floating-point collection $\{h^{(m-i)}\}$ in Algorithm 11 satisfies*

$$\left| \sum_{j=1}^k h_j^{(m-i)} - h_{m-i} \right| \leq \tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^{i-1}) H_{m-i},$$

and

$$\sum_{j=1}^k \left| h_j^{(m-i)} \right| \leq \tilde{\rho}^i H_{m-i},$$

for $i = 0, 1, \dots, m$.

Proof We proceed via induction on i . The base case, when $i = 0$, is clear. Now, suppose the result holds for some $i \geq 0$, and note that the triangle inequality implies

$$\left| \sum_{j=1}^k h_j^{(m-i-1)} - h_{m-i-1} \right| \leq \left| \sum_{j=1}^k h_j^{(m-i-1)} - \left(z \sum_{j=1}^k h_j^{(m-i)} + a_{m-i-1} \right) \right| + |z| \left| \sum_{j=1}^k h_j^{(m-i)} - h_{m-i} \right|.$$

Applying Lemma 4 and the induction hypothesis gives us

$$\begin{aligned} \left| \sum_{j=1}^k h_j^{(m-i-1)} - h_{m-i-1} \right| &\leq \tilde{\gamma}_{4k-1}^k \left(|z| \sum_{j=1}^k \left| h_j^{(m-i)} \right| + |a_{m-i-1}| \right) + |z| \tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^{i-1}) H_{m-i} \\ &\leq \tilde{\gamma}_{4k-1}^k (|z| \tilde{\rho}^i H_{m-i} + |a_{m-i-1}|) + |z| \tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^{i-1}) H_{m-i} \\ &\leq \tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^i) H_{m-i-1}. \end{aligned}$$

Moreover, by Lemma 4, we have

$$\begin{aligned} \sum_{j=1}^k \left| h_j^{(m-i-1)} \right| &\leq \tilde{\rho} \left(|z| \sum_{j=1}^k \left| h_j^{(m-i)} \right| + |a_{m-i-1}| \right) \\ &\leq \tilde{\rho} (|z| \tilde{\rho}^i H_{m-i} + |a_{m-i-1}|) \leq \tilde{\rho}^{i+1} H_{m-i-1}, \end{aligned}$$

and so the result holds for $i + 1$.

Finally, Corollary 3 gives a forward error bound and flop count for Algorithm 11. Note that when $k = 2$, the flop count is $71m + 14$, which is significantly less than the $100m - 7$ flops for the CompHorner method from [3].

Corollary 3 *Let $p \in \mathcal{C}[z]$ be a degree m polynomial, $z \in \mathcal{C}$, $4n\mu \leq 1$, and $k \geq 2$. Then, $h = \text{HornerKCmplx}(p, z, k)$ requires $m(42k^2 - 43k - 11) + (12k^2 - 22k + 10)$ flops and satisfies*

$$|h - p(z)| \leq (\mu + 3\gamma_{k-1}^2) |p(z)| + 2\tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^m) \tilde{p}(|z|).$$

Proof The flop count is left to the reader. For the error bound, note that the triangle inequality implies

$$|h - p(z)| \leq \left| h - \sum_{j=1}^k h_j^{(0)} \right| + \left| \sum_{j=1}^k h_j^{(0)} - h_0 \right|.$$

Applying Theorem 7 and Theorem 9 gives us

$$\begin{aligned} |h - p(z)| &\leq (\mu + 3\gamma_{k-1}^2) \left| \sum_{j=1}^k h_j^{(0)} \right| + \gamma_{2k-2}^k \sum_{j=1}^k |h_j^{(0)}| + \tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^{m-1}) H_0 \\ &\leq (\mu + 3\gamma_{k-1}^2) (|h_0| + \tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^{m-1}) H_0) + \tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^m) H_0 \\ &\leq (\mu + 3\gamma_{k-1}^2) |p(z)| + 2\tilde{\gamma}_{4k-1}^k (1 + \tilde{\rho} + \cdots + \tilde{\rho}^m) \tilde{p}(|z|). \end{aligned}$$

Similar to the real case, see Proposition 1, we can simplify the error bound in Corollary 3 by replacing the term $(1 + \tilde{\rho} + \cdots + \tilde{\rho}^m)$ with $(m + 8)$, for reasonably sized k and m .

Proposition 2 *For $2 \leq k \leq 10$ and $1 \leq m \leq 10^5$, we have*

$$1 + \tilde{\rho} + \cdots + \tilde{\rho}^m \leq (m + 8).$$

Proof Note that

$$1 + \tilde{\rho} + \cdots + \tilde{\rho}^m = \frac{\tilde{\rho}^{m+1} - 1}{\tilde{\rho} - 1},$$

where $\tilde{\rho} = 1 + 2\tilde{\gamma}_{4k-1} + \cdots + 2\tilde{\gamma}_{4k-1}^{k-1} + \tilde{\gamma}_{4k-1}^k$. Define $\tilde{\theta} = 2\tilde{\gamma}_{4k-1} + \cdots + 2\tilde{\gamma}_{4k-1}^{k-1} + \tilde{\gamma}_{4k-1}^k$, then $\tilde{\rho} = 1 + \tilde{\theta}$ and $\tilde{\rho}^{m+1} = \sum_{j=0}^{m+1} \binom{m+1}{j} \tilde{\theta}^j$. Now, it is true that $\binom{m+1}{j} \tilde{\theta}^j > \binom{m+1}{j+1} \tilde{\theta}^{j+1}$, for $j = 0, 1, \dots, m$; otherwise, there is a j for which $\tilde{\theta} \geq \frac{j+1}{m+1}$, which contradicts the reasonably sized k and m . Therefore, we have

$$\begin{aligned} \tilde{\rho}^{m+1} &= 1 + (m+1)\tilde{\theta} + \sum_{j=2}^{m+1} \binom{m+1}{j} \tilde{\theta}^j \\ &\leq 1 + (m+1)\tilde{\theta} + \sum_{j=2}^{m+1} \frac{(m+1)!}{2(m-1)!} \tilde{\theta}^2 \\ &= 1 + (m+1)\tilde{\theta} + \frac{m^2(m+1)}{2} \tilde{\theta}^2 \\ &\leq 1 + (m+8)\tilde{\theta}, \end{aligned}$$

where the last line follows from $m^2(m+1)\tilde{\theta}^2 \leq 14\tilde{\theta}$; otherwise, $\tilde{\theta} > \frac{14}{m^2(m+1)}$, which contradicts the reasonably sized k and m . Finally, we have

$$1 + \tilde{\rho} + \cdots + \tilde{\rho}^m = \frac{\tilde{\rho}^{m+1} - 1}{\tilde{\rho} - 1} \leq \frac{(m+8)\tilde{\theta}}{\tilde{\theta}} = (m+8).$$

Now, we can re-write the error bound from Corollary 3. Note that $3\gamma_{n-1}^2$ is negligible compared with μ . So, the result of Corollary 4 tells us that the result $h = \text{HornerKCmplx}(\mathbf{p}, z, k)$ is as accurate as if computed in k -fold precision and then rounded into the working precision.

Corollary 4 *Let $p \in \mathcal{F}[z]$ be a degree m polynomial, where $1 \leq m \leq 10^5$. Also, let $z \in \mathcal{F}$ and $2 \leq k \leq 10$. Then, $h = \text{HornerKCmplx}(p, z, k)$ satisfies*

$$|h - p(z)| \leq (\mu + 3\gamma_{k-1}^2) |p(z)| + 2(m+8)\tilde{\gamma}_{4k-1}^k \tilde{p}(|z|).$$

4 Numerical Experiments

In this section, we present the results of several numerical experiments to demonstrate the error bound and computational time of the HornerK method in Algorithm 10 and the HornerKCmplx method in Algorithm 11. Note that all higher precision computations are implemented using the GNU MPC and MPFR libraries [5,6]. All code is written in C and compiled using Apple clang version 13.0.0 and is available at <https://github.com/trcameron/HornerK>.

4.1 Error Bound

In this section, we demonstrate the relative forward error bound for HornerK, see Corollary 2, and HornerKCmplx, see Corollary 4. For HornerK, we test on the expanded form of $p_m(z) = (z - 1)^m$. Note that

$$p_m(z) = \sum_{k=0}^m \binom{m}{k} (-1)^{m-k} z^k,$$

which allows us to compute the coefficients of $p_m(z)$ exactly for reasonably sized m . Furthermore,

$$\tilde{p}_m(|z|) = \sum_{k=0}^m \binom{m}{k} |z|^k = (|z| + 1)^m,$$

which means that the condition number of $p_m(z)$ can be written as

$$\text{cond}(p_m(z)) = \frac{\tilde{p}_m(|z|)}{|p_m(z)|} = \left(\frac{|z| + 1}{|z - 1|} \right)^m$$

For the experiment reported in Figure 1, we select $z = \text{fl}(220/119)$. So, as m ranges from 2 to 50, the condition number of $p_m(z)$ ranges from 10^4 to 10^{132} . For each m , the value of $p_m(z)$ is computed with high accuracy using the MPFR library. In addition, we compute the value of $p_m(z)$ using the HornerK method, for $k = 1, 2, \dots, 8$, where $k = 1$ corresponds to the standard Horner method shown in Algorithm 9. For each k , we report the relative error in the HornerK computation, as compared to the high accuracy MPFR computation. For viewing purposes, if the relative error is less than μ then we replace its value by μ , and if the relative error is greater than 1 then we replace its value by 1. The results of this experiment are shown in Figure 1 on a log-log axis, where the x-axis corresponds to the condition number of $p_m(z)$ and the y-axis corresponds to the relative error in the computation of $p_m(z)$. This experiment clearly illustrates the result in Corollary 2. That is, the relative error in HornerK is on the order of μ until the condition number is on the order of μ^{1-k} . The tick marks on the x-axis are on the order of μ^{1-k} , for $k = 1, \dots, 8$.

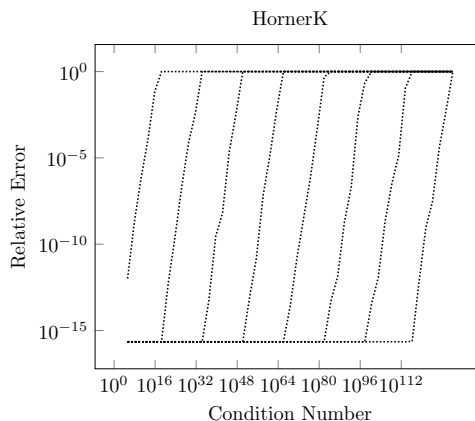


Fig. 1 Accuracy of HornerK for $k = 1, 2, \dots, 8$

For HornerKCmplx, we test on the expanded form of $p_m(z) = (z - i)^m$. Note that

$$p_m(z) = \sum_{k=0}^m \binom{m}{k} (-i)^{m-k} z^k,$$

which allows us to compute the coefficients of $p_m(z)$ exactly for reasonably sized m . Furthermore,

$$\tilde{p}_m(|z|) = \sum_{k=0}^m \binom{m}{k} |z|^k = (|z| + 1)^m,$$

which means that the condition number of $p_m(z)$ can be written as

$$\text{cond}(p_m(z)) = \frac{\tilde{p}_m(|z|)}{|p_m(z)|} = \left(\frac{|z| + 1}{|z - i|} \right)^m$$

For the experiment reported in Figure 2, we select $z = \text{fl}(220/119)i$. So, as m ranges from 2 to 50, the condition number of $p_m(z)$ ranges from 10^4 to 10^{132} . For each m , the value of $p_m(z)$ is computed with high accuracy using the MPC library. In addition, we compute the value of $p_m(z)$ using the HornerKCmplx method, for $k = 1, 2, \dots, 8$, where $k = 1$ corresponds to the standard Horner method shown in Algorithm 9. For each k , we report the relative error in the HornerKCmplx computation, as compared to the high accuracy MPC computation. For viewing purposes, if the relative error is less than $m\mu$ then we replace its value by μ , and if the relative error is greater than 1 then we replace its value by 1. The results of this experiment are shown in Figure 1 on a log-log axis, where the x-axis corresponds to the condition number of $p_m(z)$ and the y-axis corresponds to the relative error in the computation of $p_m(z)$. This experiment clearly illustrates the result in Corollary 4. That is, the relative error in HornerKCmplx is on the order of μ until the condition number is on the order of μ^{1-k} . The tick marks on the x-axis are on the order of μ^{1-k} , for $k = 1, \dots, 8$.

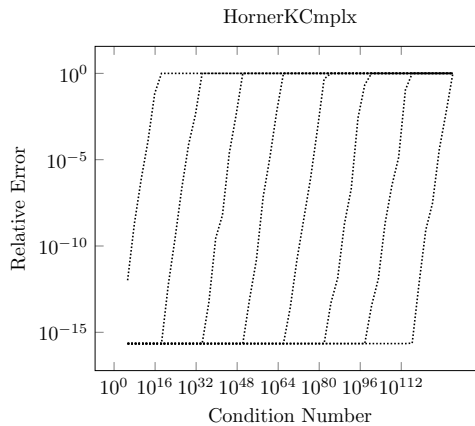


Fig. 2 Accuracy of HornerKCmplx for $k = 1, 2, \dots, 8$

4.2 Computation Time

In this section, we compare the time required for HornerK (HornerKCmplx) and MPFR (MPC) to evaluate a random polynomial at a random value. Each time MPFR (MPC) is used, the precision of the floating-point format must be specified. The IEEE 754 standard, see [1, Section 3.6], recommends that for a floating-point format of $64k$ bits, where $k \geq 2$, we use the following precision (number of bits in the significand):

$$\text{prec} = 64k - 4 \lceil \log_2(64k) \rceil + 13,$$

where $\lceil \cdot \rceil$ denotes rounding to the nearest integer.

For $k = 2, \dots, 8$, $m = 20, 40, \dots, 81920$, we construct 100 real random polynomials of degree m whose coefficients are selected from the uniform distribution $\mathcal{U}_{[-1,1]}$. We evaluate each polynomial at a random value, selected from the uniform distribution $\mathcal{U}_{[-1,1]}$, using HornerK and MPFR and we record the elapsed time for both methods. The average time for each k is reported in Table 1.

For $k = 2, \dots, 8$, $m = 20, 40, \dots, 81920$, we construct 100 complex random polynomials of degree m whose coefficients have real and imaginary parts selected from the uniform distribution $\mathcal{U}_{[-1,1]}$. We evaluate each polynomial at a random value, whose real and imaginary part is selected from the uniform distribution $\mathcal{U}_{[-1,1]}$, using HornerKCmplx and MPC and we record the elapsed time for both methods.

k/prec	2/113	3/175	4/237	5/300	6/363	7/426	8/489
HornerK	2.60E-04	4.35E-04	6.51E-04	9.86E-04	1.38E-03	1.85E-03	2.43E-03
MPFR	2.51E-03	2.62E-03	2.50E-03	2.57E-03	2.59E-03	2.62E-03	2.65E-03
Ratio	9.63	6.03	3.85	2.61	1.88	1.42	1.09

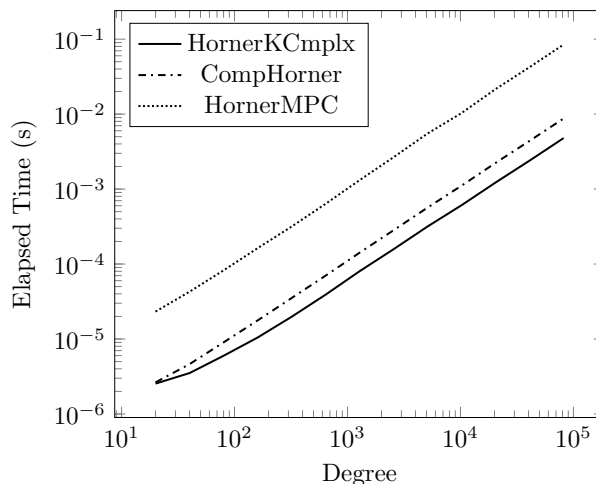
Table 1 Average elapsed time for real random polynomials.

k/prec	2/113	3/175	4/237	5/300	6/363	7/426	8/489
HornerKCmplx	7.52E-04	1.63E-03	2.96E-03	4.73E-03	6.91E-03	9.59E-03	1.25E-02
MPC	1.42E-02	1.43E-02	1.48E-02	1.56E-02	1.64E-02	1.76E-02	1.82E-02
Ratio	18.9	8.78	4.99	3.29	2.38	1.83	1.45

Table 2 Average elapsed time for complex random polynomials.

Note that the random value is normalized to avoid overflow in Horner’s method, which occurs frequently for very high-degree polynomials. The average time for each k is reported in Table 2.

Finally, we compare the elapsed time of HornerKCmplx (for $k = 2$), MPC (for $prec = 113$), and the CompHorner method from [3]. Note that all methods have a similar relative forward error bound, that is, they are as accurate as if computed in twice the working precision and then rounded into the working precision. For $m = 20, 40, \dots, 81920$, we construct 100 complex random polynomials of degree m whose coefficients have real and imaginary parts selected from the uniform distribution $\mathcal{U}_{[-1,1]}$. We evaluate each polynomial at a random value, whose real and imaginary part is selected from the uniform distribution $\mathcal{U}_{[-1,1]}$, using HornerKCmplx, CompHorner, and MPC and we record the elapsed time for all three methods. Note that the random value is normalized to avoid overflow in Horner’s method. The average elapsed time for each degree m is reported in Figure 3.

**Fig. 3** Elapsed time for complex random polynomials

5 Conclusion

The HornerK and HornerKCmplx methods are effective for the accurate evaluation of a polynomial in real and complex floating-point arithmetic, respectively. These methods are as accurate as if computed in k -fold precision and then rounded into the work precision, see Corollary 2 and Corollary 4, respectively. In Section 4, we illustrate the accuracy of both methods and demonstrate that they are significantly faster than multi-precision software MPFR and MPC, respectively, for $k \leq 8$. Moreover, when $k = 2$, we show that HornerKCmplx is faster than the CompHorner method from [3]. In future work, we will use HornerK and HornerKCmplx to derive efficient methods for computing the roots of a polynomial as accurate as if computed in k -fold precision and then rounded into the working precision.

References

1. ANSI/IEEE: IEEE Standard for Binary Floating Point Arithmetic, std 754-2008 edn. IEEE, New York, NY (2008)

2. ANSI/IEEE: IEEE Standard for Binary Floating Point Arithmetic, std 754-2019 edn. IEEE, New York, NY (2019)
3. Cameron, T.R., Graillat, S.: On a compensated Ehrlich-Aberth method for the accurate computation of all polynomial roots. *Electron. Trans. Numer. Anal.* **55**, 401–423 (2022)
4. Dekker, T.J.: A floating-point technique for extending the available precision. *Numer. Math.* **18**, 224–242 (1971)
5. Enge, A., Gastineau, M., Théveny, P., Zimmermann, P.: MPC: A library for multiprecision complex arithmetic with exact rounding. Inria, 1.2.1 edn. (2021). <http://mpc.multiprecision.org>
6. Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., Zimmermann, P.: MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.* **33**(2) (2007). <https://www.mpfr.org>
7. Gill, S.: A process for the step-by-step integration of differential equations in an automatic digital computing machine. *Proc. Cambridge Phil. Soc.* **47**, 96–108 (1951)
8. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys* **23**, 5–48 (1991)
9. Graillat, S., Louvet, N., Langlois, P.: Compensated Horner scheme. Tech. rep., Université de Perpignan Via Domitia (2005). Available at <https://www-pequan.lip6.fr/~jmc/polycopies/Compensation-horner.pdf>
10. Graillat, S., Ménessier-Morain, V.: Accurate summation, dot product and polynomial evaluation in complex floating point arithmetic. *Inform. and Comput.* pp. 57–71 (2012)
11. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. SIAM, Philadelphia, PA (2002)
12. Kahan, W.: Further remarks on reducing truncation errors. *Comm. ACM* **8**, 40 (1965)
13. Knuth, D.E.: The Art of Computer Programming: Seminumerical Algorithms, vol. 2. Addison-Wesley, Reading, MA (1998)
14. Langlois, P., Louvet, N.: Compensated Horner algorithm in K times the working precision. Research Report (2008). Available at <https://hal.inria.fr/inria-00267077>
15. Møller, O.: Quasi double-precision in floating point addition. *BIT* **5**, 37–50 (1965)
16. Nievergelt, Y.: Scalar fused multiply-add instructions produce floating-point matrix arithmetic provably accurate to the penultimate digit. *ACM Trans. Math. Softw.* **29**(1), 27–48 (2003)
17. Ogita, T., Rump, S.M., Oishi, S.: Accurate sum and dot product. *SIAM J. Sci. Comput.* **26**(6), 1955–1988 (2005)
18. Rump, S.M.: Inversion of extremely ill-conditioned. *Japan J. Indust. Appl. Math.* **26**, 249–277 (2009)