mathematics

Article

Accurate Goertzel Algorithm: Error Analysis, Validations and Applications

Chuanying Li [1], Peibing Du [2,*], Kuan Li [3], Yu Liu [2], Hao Jiang [4] and Zhe Quan [1]

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China; lichuanying@hnu.edu.cn (C.L.); quanzhe@hnu.edu.cn (Z.Q.)
[2] Northwest Institute of Nuclear Technology, Xi'an 710024, China; liuyu05@nint.ac.cn
[3] School of Cyberspace Security, Dongguan University of Technology, Dongguan 523106, China; likuan@dgut.edu.cn
[4] College of Computer, National University of Defense Technology, Changsha 410073, China; haojiang@nudt.edu.cn
[*] Correspondence: dupeibing1@nint.ac.cn

Abstract: The Horner and Goertzel algorithms are frequently used in polynomial evaluation. Each of them can be less expensive than the other in special cases. In this paper, we present a new compensated algorithm to improve the accuracy of the Goertzel algorithm by using error-free transformations. We derive the forward round-off error bound for our algorithm, which implies that our algorithm yields a full precision accuracy for polynomials that are not too ill-conditioned. A dynamic error estimate in our algorithm is also presented by running round-off error analysis. Moreover, we show the cases in which our algorithms are less expensive than the compensated Horner algorithm for evaluating polynomials. Numerical experiments indicate that our algorithms run faster than the compensated Horner algorithm in those cases while producing the same accurate results, and our algorithm is absolutely stable when the condition number is smaller than 10^{16}. An application is given to illustrate that our algorithm is more accurate than MATLAB's `fft` function. The results show that the relative error of our algorithm is from 10^{15} to 10^{17}, and that of the `fft` was from 10^{12} to 10^{15}.

Keywords: polynomial evaluation; goertzel algorithm; round-off error; error-free transformation; compensated algorithm; numerical stability

MSC: 68U01

Citation: Li, C.; Du, P.; Li, K.; Liu, Y.; Jiang, H.; Quan, Z. Accurate Goertzel Algorithm: Error Analysis, Validations and Applications. *Mathematics* **2022**, *10*, 1788. https://doi.org/10.3390/math10111788

Academic Editors: Theodore E. Simos and Charampos Tsitouras

Received: 13 April 2022
Accepted: 18 May 2022
Published: 24 May 2022

1. Introduction

Polynomial evaluation is ubiquitous in computational sciences and their applications, such as interpolation and approximation practices and signal processing. This article will investigate a broader situation of polynomial evaluation:

$$\omega(z) = \sum_{n=0}^{N} a_n z^n, \tag{1}$$

where $z, a_0, a_1, \ldots, a_N \in \mathbb{C}$. The nested-type algorithms are usually used to evaluate polynomials. The Horner algorithm is the most widely used polynomial evaluation algorithm [1]. In special cases, like $z \in \mathbb{C}$ and $a_0, a_1, \ldots, a_N \in \mathbb{R}$, the Goertzel algorithm that can be applied to compute the discrete Fourier transform (DFT) of specific indices in a vector [2,3] is less expensive the Horner algorithm. The numerical stability of the Horner and Goertzel algorithms was given by Wilkinson [4] and Smoktunowicz [5]. The computed results from these algorithms are arbitrarily less accurate than the working precision u when the polynomial is ill-conditioned due to the round-off errors in floating-point arithmetic. The relative accuracy of these algorithms verifies the following priori bound:

$$\frac{|\omega(z) - \widehat{\omega}(z)|}{|\omega(z)|} \leq \texttt{cond}(\omega, z) \times \mathcal{O}(u), \tag{2}$$

where $\widehat{\omega}(z)$ is the computed result and $\texttt{cond}(\omega, z) = \sum_{n=0}^{N} |a_n||z|^n / |\sum_{n=0}^{N} a_n z^n|$ is the condition number.

In order to improve the accuracy of double precision, Bailey [6] proposed a famous library for double-double and quad-double arithmetic. However, this library needs to normalize floating-point numbers in every operation, and thus the instruction level parallelism is affected [7,8]. The compensated algorithm is improved to solve this problem with the developments and applications of error-free transformation [9]. The relative accuracy of compensated algorithms verifies the following priori bound:

$$\frac{|\omega(z) - \overline{\omega}(z)|}{|\omega(z)|} \leq u + \texttt{cond}(\omega, z) \times \mathcal{O}(u^2), \tag{3}$$

where $\overline{\omega}(z)$ is the computed result of a compensated algorithm.

Recently, compensated algorithms have been widely studied in evaluating polynomials. Graillat [10] proposed a compensated Horner algorithm that achieves full precision accuracy for polynomials that are not excessively ill-conditioned. Aside from that, he extended the error-free transformation and compensated Horner algorithm in complex floating-point arithmetic [11,12] and applied a compensated Horner algorithm to evaluate rational functions [13] and solve all polynomial roots [14]. Polynomial series represented in other basis were also considered, such as the Chebyshev form evaluated by a compensated Chenshaw algorithm [15], the Bernstein form evaluated by a compensated de Casteljau algorithm [16], and a compensated Volk and Schumaker(VS) algorithm [17]. Furthermore, the compensated idea is also applied to matrix multiplication to obtain more accurate results [18–20].

With the wide application of floating-point numbers and floating-point operations in numerical computing, the analysis of rounding errors has become the focus [21,22]. Running round-off errors are analyzed and applied to many algorithms of polynomial evaluation [23]. Delgado [17] proposed an adaptive evaluation algorithm by using the de Casteljau algorithm and compensated VS algorithms with a dynamic error estimate. Jiang [24] presented running round-off error analysis for evaluating elementary symmetric functions in real and complex floating-point arithmetic. Barrio [25] developed a more complete compensated algorithm library to evaluate orthogonal polynomial series with dynamic error estimates. In addition, error analysis can also be used for machine learning and the numerical solution of differential equations [26].

In this paper, our contributions are as follows:

- We design a new compensated Goertzel algorithm and prove that our algorithm can almost yield full working precision to evaluate polynomials (1);
- We propose dynamic error estimates, which can offer a sharper bound for our approach without considerably increasing its computing complexity;
- Numerical experiments show that our algorithm runs faster than the compensated Horner algorithm in some cases while keeping a similar precision accuracy;
- An application is given to illustrate that our algorithm outperforms MATLAB's `fft` when dealing with the DFT.

The rest of this paper is organized as follows. Section 2 introduces our compensated Goertzel algorithm. A dynamic error estimate is proposed in Section 3. Section 4 analyzes numerical experiment results and gives an application to illustrate that our algorithm outperforms them. Finally, the full paper is summarized in Section 5.

2. Goertzel Compensated Algorithm

We assume working with IEEE-754 floating-point standard [27] rounding to the nearest value in this paper. Let $\mathbb{F}$ be the set of floating-point numbers, $\mathbb{C}$ represent the complex number, and $\{\oplus, \ominus, \otimes, \oslash\}$ represent a floating-point operation. This part presents how to design the Goertzel compensated algorithm. First, the Goertzel algorithm and its relationship with the Clenshaw algorithm are listed. Then, the error-free transformations and sum of squares algorithm are recalled. At last, we present the compensated Goertzel algorithm.

2.1. Goertzel Algorithm

By assuming $\lambda, z \in \mathbb{C}$ and $z = x + iy$, then we have a quadratic polynomial

$$(\lambda - z)(\lambda - \bar{z}) = \lambda^2 - p\lambda + q, \tag{4}$$

where $p = 2x$ and $q = |z|^2$. By dividing the polynomial in Equation (1) by that in Equation (4), we obtain

$$\omega(\lambda) = b_0 + b_1\lambda + (\lambda - z)(\lambda - \bar{z}) \sum_{n=2}^{N} b_n \lambda^{n-2}, \tag{5}$$

where

$$\begin{cases} a_0 = b_0 + qb_2, \\ a_1 = b_1 - pb_2 + qb_3, \\ \quad \vdots \\ a_n = b_n - pb_{n+1} + qb_{n+2}. \end{cases} \tag{6}$$

Thus, the evaluated result of the polynomial in Equation (1) is

$$\omega(z) = b_0 + b_1 z. \tag{7}$$

Above Equations (5)–(7), we can find the Goertzel algorithm [2] with Algorithm 1.

Algorithm 1 Polynomial evaluation by Goertzel algorithm

```
**Function :** $\omega(z) = \texttt{Goertzel}((a_n)_{n=0}^N, z)$
**Require :** $z = x + iy \in \mathbb{C}$, $(a_n)_{n=0}^N \in \mathbb{C}$
**Ensure :** $\omega(z) = \sum_{n=0}^N a_n z^n$
  $p = 2x, q = x^2 + y^2$
  $b_{N+1} = b_{N+2} = 0$
  for $n = N, N-1, ..., 1$
    $b_n = a_n + pb_{n+1} - qb_{n+2}$
  end
  $b_0 = a_0 + xb_1 - qb_2$
  $\omega(z) = b_0 + iyb_1$
```

In floating-point arithmetic, a backward error bound for the computed result of the polynomial evaluation by Algorithm 1 is presented by Smoktunowicz [5] as Theorem 1:

Theorem 1. *Assume $z, a_n \in \mathbb{F} + i\mathbb{F}$ for $n = 0, \ldots, N$. Let $10N^2u \leq 0.1$. Then the Goertzel algorithm for evaluating the polynomial in Equation (1) is componentwise backward stable such that*

$$\widehat{\omega}(z) = \sum_{n=0}^{N} a_n(1 + \Delta_n)z^n, \tag{8}$$

where

$$|\Delta_n| \leq 10N^2u + \mathcal{O}(u^2). \tag{9}$$

In fact, Algorithm 1 is a special case of the Clenshaw algorithm [2,5]. When we let $t = x/|z|$ and $B_n = b_n|z|^n$, Algorithm 1 can be represented in Clenshaw form [28]:

$$B_n = a_n|z|^n + 2tB_{n+1} - B_{n+2}. \tag{10}$$

According to the properties of the Chebyshev polynomial series [29] evaluated by the Clenshaw algorithm, we have

$$\begin{cases} b_n = \sum_{k=n}^{N} a_k|z|^{k-n}U_{k-n}(t), \\ b_0 = \sum_{k=0}^{N} a_k|z|^k T_k(t), \\ n = 1, 2, \dots, N. \end{cases} \tag{11}$$

where $T_k(t)$ and $U_k(t)$ are Chebyshev polynomials of the first and second kinds, respectively. They satisfy

$$|T_k(t)| \le 1, |U_k(t)| \le k+1, \tag{12}$$

and

$$\frac{z^k}{|z|^k} = T_k(t) + i\frac{y}{|z|}U_{k-1}(t) \quad for \quad k = 0, 1, \dots, N. \tag{13}$$

2.2. Error-Free Transformations and Sum of Squares Algorithm

The basic algorithms of error-free transformations are TwoSum and TwoProd, which were presented by Knuth [30] and Dekker [31], respectively. Graillat [11] extended the addition and multiplication to complex number cases, which are TwoSumCplx and TwoProdCplx, respectively. In this paper, we shall use a new product error-free transformation of one real and one complex floating-point number, which is called TwoProdRC in Algorithm 2.

Algorithm 2 Error-free transformation of the product of real and complex floating-point numbers

```
Function :$[x, y] = \texttt{TwoProdRC}(a, b)$
Require : $a \in \mathbb{F}, b = c + id \in \mathbb{F} + i\mathbb{F}$
Ensure : $x + y = a \times b$
  $[p, e] = \texttt{TwoProd}(a, c)$
  $[f, g] = \texttt{TwoProd}(a, d)$
  $x = p + if$
  $y = e + ig$
```

The details of the error-free transformations above are presented in Table 1.

Table 1. Error-free transformations, their properties and operation costs.

Algorithm	Properties	Flops
$[x, y] = \texttt{TwoSum}(a, b)$	$x = a \oplus b, x + y = a + b$	6
$[x, y] = \texttt{TwoProd}(a, b)$	$x = a \otimes b, x + y = a \times b$	17
$[x, y] = \texttt{TwoProdRC}(a, b)$	$x = a \otimes b, x + y = a \times b$	34
$[x, y] = \texttt{TwoSumCplx}(a, b)$	$x = a \oplus b, x + y = a + b$	12
$[p, e, f, g] = \texttt{TwoProdCplx}(a, b)$	$p = a \otimes b, p + e + f + g = a \times b$	80

$\{\oplus, \otimes\}$ represents $\{+, \times\}$ in floating-point operations.

Furthermore, the sum of squares algorithm [32] is given in Algorithm 3. It requires 42 flops.

Algorithm 3 Sum of squares by two floating-point numbers

```
Function : $[x, y] = \texttt{SumOfSquares}(a, b)$
Require : $a, b \in \mathbb{F}$
Ensure : $x + y \approx a^2 + b^2$
  $[p, f] = \texttt{TwoProd}(a, a)$
  $[e, g] = \texttt{TwoProd}(b, b)$
  $[x, h] = \texttt{TwoSum}(p, e)$
  $y = f \oplus g \oplus h$
```

2.3. Compensated Goertzel Algorithm

Although Algorithm 1 is a special Clenshaw algorithm, computation via Equation (10) is more expensive. Thus, using the compensated Clenshaw algorithm [28] to express the compensated Goertzel algorithm is not a good idea. We design a compensated Goertzel algorithm by using `SumOfSquares`, `TwoSumCplx` and `TwoProdRC` algorithms to record the round-off errors.

Assume $\widehat{a} \in \mathbb{F}$ is a computed result in floating-point arithmetic, and its perturbation is ϵa such that

$$\widehat{a} = a + \epsilon a. \tag{14}$$

`SumOfSquares` can find the approximate round-off error $\widehat{\epsilon q}$ of q. Other round-off errors in Algorithm 1 can be accurately computed by error-free transformation algorithms `TwoSumCplx` and `TwoProdRC`. Then, we can combine all round-off errors by $\widehat{\ell}_n$ and find the approximate perturbation $\widehat{\epsilon b}_n$ of b_n for $n = N-1, N-2, \ldots, 1$ in each loop. The loop ends using `TwoProdRC` and `TwoSumCplx` to calculate $\widehat{b}_0$ and combines all round-off errors to obtain the approximate perturbation. The round-off error of $y \otimes \widehat{b}_1$ should also be considered by `TwoProdRC`. The compensated Goertzel algorithm is presented in Algorithm 4, and Figure 1 shows the flow chart of this algorithm.

Algorithm 4 Polynomial evaluation by compensated Goertzel algorithm

```
Function: $\overline{\omega}(z) = \texttt{CompGoertzel}((a_n)_{n=0}^{N}, z)$
Require: $z = x + iy \in \mathbb{F} + i\mathbb{F}$, $(a_n)_{n=0}^{N} \in \mathbb{F} + i\mathbb{F}$
Ensure: $\overline{\omega}(z) \approx \sum_{n=0}^{N} a_n z^n$
  $p = 2x$
  $[\widehat{q}, \widehat{\epsilon q}] = \texttt{SumOfSquares}(x, y)$
  $\widehat{b}_N = a_N, \widehat{b}_{N+1} = \widehat{\epsilon b}_N = \widehat{\epsilon b}_{N+1} = 0$
  for $n = N-1, N-2 \ldots, 1$
    $[r_n, \pi_n] = \texttt{TwoProdRC}(p, \widehat{b}_{n+1})$
    $[s_n, \sigma_n] = \texttt{TwoProdRC}(-\widehat{q}, \widehat{b}_{n+2})$
    $[t_n, \eta_n] = \texttt{TwoSumCplx}(r_n, s_n)$
    $[\widehat{b}_n, \xi_n] = \texttt{TwoSumCplx}(t_n, a_n)$
    $\widehat{\ell}_n = \pi_n \oplus \sigma_n \oplus \eta_n \oplus \xi_n \ominus \widehat{\epsilon q} \otimes \widehat{b}_{n+2}$
    $\widehat{\epsilon b}_n = \widehat{\ell}_n \oplus p \otimes \widehat{\epsilon b}_{n+1} \ominus \widehat{q} \otimes \widehat{\epsilon b}_{n+2}$
  end
  $[r_0, \pi_0] = \texttt{TwoProdRC}(x, \widehat{b}_1)$
  $[s_0, \sigma_0] = \texttt{TwoProdRC}(-\widehat{q}, \widehat{b}_2)$
  $[t_0, \eta_0] = \texttt{TwoSumCplx}(r_0, s_0)$
  $[\widehat{b}_0, \xi_0] = \texttt{TwoSumCplx}(t_0, a_0)$
  $\widehat{\ell}_0 = \pi_0 \oplus \sigma_0 \oplus \eta_0 \oplus \xi_0 \ominus \widehat{\epsilon q} \otimes \widehat{b}_2$
  $\widehat{\epsilon b}_0 = \widehat{\ell}_0 \oplus x \otimes \widehat{\epsilon b}_1 \ominus \widehat{q} \otimes \widehat{\epsilon b}_2$
  $[\phi, \psi] = \texttt{TwoProdRC}(y, \widehat{b}_1)$
  $\widehat{e} = \widehat{\epsilon b}_0 \oplus i(\widehat{\epsilon b}_1 \otimes y \oplus \psi)$
  $\widehat{\omega}(z) = \widehat{b}_0 \oplus i\phi$
  $\overline{\omega}(z) = \widehat{\omega}(z) \oplus \widehat{e}$
```

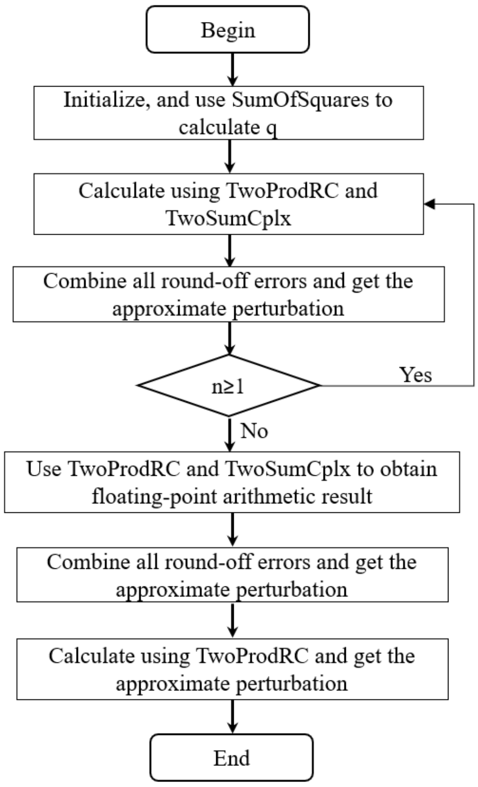
Begin
Initialize, and use SumOfSquares to calculate q
Calculate using TwoProdRC and TwoSumCplx
Combine all round-off errors and get the approximate perturbation
n≥1
Yes
No
Use TwoProdRC and TwoSumCplx to obtain floating-point arithmetic result
Combine all round-off errors and get the approximate perturbation
Calculate using TwoProdRC and get the approximate perturbation
End

Figure 1. The flowchart of the compensated Goertzel algorithm.

We remark that if $(a_n)_{n=0}^{N} \in \mathbb{F}$, then we shall replace `TwoSumCplx` and `TwoProdRC` with `TwoSum` and `TwoProd` in Algorithm 4, respectively.

3. Round-Off Error and Complexity Analysis

In this section, we consider the error bound and complexity of Algorithm 4 through Higham's theories [33]. In our analysis, we assume that there is no computational overflow or underflow. First, we present the priori bound by forward round-off error analysis. Then, we show a dynamic error estimate by running round-off error analysis. At last, we compare the complexities of Horner, Goertzel, their compensated algorithms and the compensated Goertzel algorithm with a dynamic error estimate to evaluate the polynomial in Equation (1) in real and complex coefficients.

3.1. Forward Round-Off Error Analysis

Let $\blacklozenge \in \{\oplus, \ominus, \otimes, \oslash\}$, $\circ \in \{+, -, \times, \div\}$ and $a, b \in \mathbb{F}$. Then, a floating-point computation obeys the model

$$a \blacklozenge b = (a \circ b)(1 + \varepsilon_1) = \frac{a \circ b}{1 + \varepsilon_2}, \tag{15}$$

where $|\varepsilon_1|, |\varepsilon_2| \leq u$. We define

$$<N> := 1 + \theta_N = \prod_{n=1}^{N} (1 + \varepsilon_n)^{\rho_n}, \tag{16}$$

where $|\varepsilon_n| \leq u, \rho_n = \pm 1$ and

$$|\theta_n| \leq \gamma_n := \frac{nu}{(1 - nu)}, \tag{17}$$

for $n = 1, 2, \ldots, N$ and $Nu < 1$. We assume that $\widehat{a}$ and $\widehat{b}$ in real arithmetic are denoted by $\widetilde{c} = \widehat{a} \circ \widehat{b}$, which will be used in our later analysis.

Lemma 1 summarizes the properties of the error-free transformations in Table 1:

Lemma 1. *For $a, b, x, y \in \mathbb{F}$, $[x, y] = \texttt{TwoSum}(a, b)$ verifies*

$$|y| \le u|x|, \quad |y| \le u|a+b|, \tag{18}$$

In addition, $[x, y] = \texttt{TwoProd}(a, b)$ verifies

$$|y| \le u|x|, \quad |y| \le u|a \times b|. \tag{19}$$

For $a, b, x, y \in \mathbb{F} + i\mathbb{F}$, $[x, y] = \texttt{TwoSumCplx}(a, b)$ verifies

$$|y| \le u|x|, \quad |y| \le u|a+b|, \tag{20}$$

Additionally, $[x, y] = \texttt{TwoProdRC}(a, b)$ verifies

$$|y| \le u|x|, \quad |y| \le u|a \times b|. \tag{21}$$

For $a, b, p, e, f, g \in \mathbb{F} + i\mathbb{F}$, $[p, e, f, g] = \texttt{TwoProdCplx}(a, b)$ verifies

$$|e + f + g| \le \sqrt{2}\gamma_2 |a \times b|. \tag{22}$$

Proof. Equations (18)–(22) are presented in [9,11]. In Algorithm 2, it is easy to obtain Equation (21) from Equation (18). □

Lemma 2 shows the property of Algorithm 3:

Lemma 2. *For $a, b, x, y \in \mathbb{F}$, $[x, y] = \texttt{SumOfSquares}(a, b)$ verifies*

$$\begin{aligned} &|x| \le (1+\gamma_2)|a^2 + b^2|, \\ &|y| \le \gamma_2 |x|, \\ &|x + y - (a^2 + b^2)| \le u\gamma_3 |a^2 + b^2|. \end{aligned} \tag{23}$$

Proof. According to Algorithm 3 and Table 1, $x = a \otimes a \oplus b \otimes b$. With Lemma 1, we obtain $|f| \le u|p|$, $|g| \le u|e|$ and $|h| \le u|x|$. From Equations (15)–(17), we have

$$\begin{aligned} |x| &= |a^2<2> + b^2<2>| \le (1+\gamma_2)|a^2 + b^2|, \\ |y| &= |f<1> + g<2> + h<1>| \le (1+\gamma_2)(|f| + |g| + |h|) \\ &\le u(1+\gamma_2)(|p| + |e| + |x|) \le 2u(1+\gamma_2)|x| = \gamma_2|x|. \end{aligned} \tag{24}$$

From Theorem 4.2 in [32], $|x + y - (a^2 + b^2)| \le \frac{3u^2}{1-3u^2}|a^2 + b^2| \le u\gamma_3|a^2 + b^2|$. □

Theorem 2 presents a priori error bound of Algorithm 4 to evaluate the polynomial in Equation (1) in complex floating-point arithmetic.

Theorem 2. *Assume $z, a_n \in \mathbb{F} + i\mathbb{F}$ for $n = 0, \dots, N$. Then, the relative forward round-off error bound in the compensated Goertzel algorithm for evaluting $\omega(z) = \sum_{n=0}^{N} a_n z^n$ in floating-point arithmetic satisfies*

$$\frac{|\texttt{CompGoertzel}((a_n)_{n=0}^{N}, z) - \omega(z)|}{|\omega(z)|} \le u + 3N^2 \gamma_{15}\gamma_{3N+1}\texttt{cond}(\omega, z). \tag{25}$$

Proof. Assume the error of $\widehat{\omega}(z)$ is e (i.e., $\widehat{\omega}(z) + e = \omega(z)$). Then, we have

$$\begin{aligned} |\overline{\omega}(z) - \omega(z)| &= |(\widehat{\omega}(z) \oplus \widehat{e}) - \omega(z)| = |(1+\varepsilon)(\widehat{\omega}(z) + \widehat{e}) - \omega(z)| \\ &= |(1+\varepsilon)(\omega(z) - e + \widehat{e}) - \omega(z)| \le u|\omega(z)| + (1+u)|e - \widehat{e}|, \end{aligned} \tag{26}$$

and

$$e = \epsilon b_0 + i(\epsilon b_1 y + \psi), \tag{27}$$

where ϵb_0 and ϵb_1 are the error of b_0 and b_1, respectively, such that

$$\begin{aligned} \epsilon b_n &= \ell_n + p\epsilon b_{n+1} + q\epsilon b_{n+2}, \\ \epsilon b_0 &= \ell_0 + x\epsilon b_1 + q\epsilon b_2, \end{aligned} \tag{28}$$

where $\ell_n = \pi_n + \sigma_n + \eta_n + \xi_n - \epsilon q b_{n+2}$ for $n = N-1, N-2\ldots, 0$. Similarly, let

$$\begin{aligned} \widetilde{e} &= \widetilde{\epsilon b}_0 + i(\widetilde{\epsilon b}_1 y + \psi), \\ \widehat{e} &= \widehat{\epsilon b}_0 \oplus i(\widehat{\epsilon b}_1 \otimes y \oplus \psi), \end{aligned} \tag{29}$$

where

$$\begin{aligned} \widetilde{\epsilon b}_n &= \widehat{\ell}_n + p\widetilde{\epsilon b}_{n+1} + \widehat{q}\widetilde{\epsilon b}_{n+2}, \\ \widetilde{\epsilon b}_0 &= \widehat{\ell}_0 + x\widetilde{\epsilon b}_1 + \widehat{q}\widetilde{\epsilon b}_2, \end{aligned} \tag{30}$$

and

$$\begin{aligned} \widehat{\epsilon b}_n &= \widehat{\ell}_n \oplus p \otimes \widehat{\epsilon b}_{n+1} \oplus \widehat{q} \otimes \widehat{\epsilon b}_{n+2}, \\ \widehat{\epsilon b}_0 &= \widehat{\ell}_0 \oplus x \otimes \widehat{\epsilon b}_1 \oplus \widehat{q} \otimes \widehat{\epsilon b}_2, \end{aligned} \tag{31}$$

where $\widehat{\ell}_n = \pi_n \oplus \sigma_n \oplus \eta_n \oplus \xi_n \ominus \widehat{\epsilon q} \otimes \widehat{b}_{n+2}$ for $n = N-1, N-2\ldots, 0$. Then, Equation (26) can be simplified as

$$|\overline{\omega}(z) - \omega(z)| \leq u|\omega(z)| + (1+u)(|e - \widetilde{e}| + |\widetilde{e} - \widehat{e}|). \tag{32}$$

First, we consider the bound of $|e - \widetilde{e}|$. From Equations (11), (28) and (30), we have

$$\begin{aligned} \epsilon b_1 &= \sum_{n=1}^{N} \ell_n |z|^{n-1} U_{n-1}(t), \\ \epsilon b_0 &= \sum_{n=0}^{N} \ell_n |z|^{n} T_n(t), \\ \widetilde{\epsilon b}_1 &= \sum_{n=1}^{N} \widehat{\ell}_n |z|^{n-1} U_{n-1}(t), \\ \widetilde{\epsilon b}_0 &= \sum_{n=0}^{N} \widehat{\ell}_n |z|^{n} T_n(t). \end{aligned} \tag{33}$$

Then, by Equations (13), (27) and (29), we deduce

$$|e - \widetilde{e}| = |(\epsilon b_0 - \widetilde{\epsilon b}_0) + i(\epsilon b_1 - \widetilde{\epsilon b}_1)y| = |\sum_{n=0}^{N-1} (\ell_n - \widehat{\ell}_n) z^n| \leq \sum_{n=0}^{N-1} |\ell_n - \widehat{\ell}_n||z|^n. \tag{34}$$

In Algorithm 4, according to Lemmas 1 and 2, we obtain

$$\begin{aligned} |\pi_n| &\leq u|p\widehat{b}_{n+1}| \leq 2u|z||\widehat{b}_{n+1}|, \\ |\sigma_n| &\leq u|\widehat{q}\widehat{b}_{n+2}| \leq u(1+\gamma_2)|z|^2|\widehat{b}_{n+2}|, \\ |\eta_n| &\leq u|p\widehat{b}_{n+1} - \widehat{q}\widehat{b}_{n+2}| \leq u(1+\gamma_2)(2|z||\widehat{b}_{n+1}| + |z|^2|\widehat{b}_{n+2}|), \\ |\xi_n| &\leq u|a_n + p\widehat{b}_{n+1} - \widehat{q}\widehat{b}_{n+2}| \leq u(1+\gamma_2)(|a_n| + 2|z||\widehat{b}_{n+1}| + |z|^2|\widehat{b}_{n+2}|), \\ |\widehat{\epsilon q}| &\leq \gamma_2|\widehat{q}| \leq \gamma_2(1+\gamma_2)|z|^2, \end{aligned} \tag{35}$$

for $n = 0, 1, \ldots, N$, and then

$$\begin{aligned} &|\pi_n| + |\sigma_n| + |\eta_n| + |\xi_n| + |\widehat{\epsilon q}||\widehat{b}_{n+2}| \\ \leq &(3u + \gamma_2)(1 + \gamma_2)(|a_n| + 2|z||\widehat{b}_{n+1}| + |z|^2|\widehat{b}_{n+2}|). \end{aligned} \tag{36}$$

In Algorithm 4, from Equations (15)–(17), we have

$$\begin{cases} \widehat{b}_{N-1} = pa_N\langle 2\rangle + a_{N-1}\langle 1\rangle, \\ \widehat{b}_{N-2} = p\widehat{b}_{N-1}\langle 3\rangle - qa_N\langle 5\rangle + a_{N-2}\langle 1\rangle \\ \qquad = a_N(p^2 - q)\langle 5\rangle + pa_{N-1}\langle 3\rangle + a_{N-2}\langle 1\rangle, \\ \quad \vdots \\ \widehat{b}_1 = p\widehat{b}_2\langle 3\rangle - q\widehat{b}_3\langle 5\rangle + a_1\langle 1\rangle \\ \qquad = a_N(p^{N-1} + \cdots)\langle 3N-4\rangle + a_{N-1}(p^{N-2} + \cdots)\langle 3N-5\rangle + \cdots, \\ \widehat{b}_0 = x\widehat{b}_1\langle 3\rangle - \widehat{q}\widehat{b}_2\langle 5\rangle + a_0\langle 1\rangle \\ \qquad = a_N(p^N + \cdots)\langle 3N-1\rangle + a_{N-1}(p^{N-1} + \cdots)\langle 3N-2\rangle + \cdots, \end{cases} \tag{37}$$

Then, by induction, we get

$$|b_n - \widehat{b}_n| \leq \gamma_{3(N-n)-1}|b_n|, \tag{38}$$

and

$$|\widehat{b}_n| \leq (1 + \gamma_{3(N-n)-1})|b_n|. \tag{39}$$

Assume that

$$g_k = \sum_{n=k}^{N} |a_n||z|^{n-k}, k = 0, 1, \ldots, N, \tag{40}$$

Given this, then

$$\sum_{n=0}^{N} g_n|z|^n \leq (N+1)g_0. \tag{41}$$

By combining Equations (11), (12) and (40), we have

$$\begin{aligned} &|b_n| \leq (N - n + 1)g_n, n = 1, \ldots, N. \\ &|b_0| \leq g_0. \end{aligned} \tag{42}$$

It is easy to obtain that

$$\widehat{\ell}_n = \pi_n\langle 4\rangle + \sigma_n\langle 4\rangle + \eta_n\langle 3\rangle + \xi_n\langle 2\rangle - \widehat{\epsilon q}\widehat{b}_{n+2}\langle 2\rangle,$$

Through Lemma 2 and Equations (36)–(42), we deduce

$$\begin{aligned} &|\ell_n - \widehat{\ell}_n| \leq \gamma_4(|\pi_n| + |\sigma_n| + |\eta_n| + |\xi_n| + |\widehat{\epsilon q}||\widehat{b}_{n+2}|) + |\epsilon q b_{n+2} - \widehat{\epsilon q}\widehat{b}_{n+2}| \\ \leq &\gamma_4(3u + \gamma_2)(1 + \gamma_2)(|a_n| + 2|z||\widehat{b}_{n+1}| + |z|^2|\widehat{b}_{n+2}|) + |\epsilon q||b_{n+2} - \widehat{b}_{n+2}| \\ &+ |\epsilon q - \widehat{\epsilon q}||\widehat{b}_{n+2}| \\ \leq &\gamma_4\gamma_5(|a_n| + 2|z||\widehat{b}_{n+1}| + |z|^2|\widehat{b}_{n+2}|) + \gamma_2\gamma_{3(N-n)-7}|z|^2|b_{n+2}| + u\gamma_3|z|^2|\widehat{b}_{n+2}| \\ \leq &[\gamma_5^2(1 + \gamma_{3(N-n)-4}) + \gamma_2\gamma_{3(N-n)-7}](|a_n| + 2|z||b_{n+1}| + |z|^2|b_{n+2}|) \\ \leq &\gamma_5\gamma_{3(N-n)+1}(|a_n| + 2|z||b_{n+1}| + |z|^2|b_{n+2}|) \\ \leq &\gamma_5\gamma_{3(N-n)+1}(|a_n| + 2(N-n)|z|g_{n+1} + (N-n-1)|z|^2g_{n+2}) \\ \leq &3\gamma_5\gamma_{3(N-n)+1}(N-n)g_n. \end{aligned} \tag{43}$$

Thus, from Equations (34), (41) and (43), we obtain

$$|e - \widetilde{e}| \leq \sum_{n=0}^{N-1} |\ell_n - \widehat{\ell}_n||z|^n \leq 3N^2\gamma_5\gamma_{3N+1}g_0. \tag{44}$$

Second, we consider the bound of $|\widetilde{e} - \widehat{e}|$. In Algorithm 4, according to Equations (15)–(17), we have

$$\begin{cases} \widehat{\epsilon b}_{N-1} = \widehat{\ell}_{N-1}, \\ \widehat{\epsilon b}_{N-2} = p\widehat{\ell}_{N-1}\langle 2\rangle + \widehat{\ell}_{N-2}\langle 1\rangle, \\ \widehat{\epsilon b}_{N-3} = p\widehat{\epsilon b}_{N-2}\langle 3\rangle - \widehat{q}\widehat{\ell}_{N-1}\langle 3\rangle + \widehat{\ell}_{N-3}\langle 1\rangle \\ \qquad = \widehat{\ell}_{N-1}(p^2 - \widehat{q})\langle 5\rangle + p\widehat{\ell}_{N-2}\langle 3\rangle + \widehat{\ell}_{N-3}\langle 1\rangle, \\ \qquad \vdots \\ \quad \widehat{\epsilon b}_1 = p\widehat{\epsilon b}_2\langle 3\rangle - \widehat{q}\widehat{\epsilon b}_3\langle 3\rangle + \widehat{\ell}_1\langle 1\rangle \\ \qquad = \widehat{\ell}_{N-1}(p^{N-2} + \cdots)\langle 3N-7\rangle + \widehat{\ell}_{N-2}(p^{N-3} + \cdots)\langle 3N-8\rangle + \cdots, \\ \quad \widehat{\epsilon b}_0 = x\widehat{\epsilon b}_1\langle 3\rangle - \widehat{q}\widehat{\epsilon b}_2\langle 3\rangle + \widehat{\ell}_0\langle 1\rangle \\ \qquad = \widehat{\ell}_{N-1}(p^{N-1} + \cdots)\langle 3N-4\rangle + \widehat{\ell}_{N-2}(p^{N-2} + \cdots)\langle 3N-5\rangle + \cdots. \end{cases} \tag{45}$$

Then, by induction, from Equations (13) and (33) as well as Lemma 1, we deduce

$$|(\widetilde{\epsilon b}_0 - \widehat{\epsilon b}_0) + i(\widetilde{\epsilon b}_1 - \widehat{\epsilon b}_1)y| \leq \gamma_{3N-4}|\sum_{n=0}^{N-1} \widehat{\ell}_n z^n| \leq \gamma_{3N-4}\sum_{n=0}^{N-1} |\widehat{\ell}_n||z|^n, \tag{46}$$

and

$$\begin{aligned} |\widehat{\epsilon b}_0 + i(\widehat{\epsilon b}_1 y + \psi)| &\leq |\widehat{\epsilon b}_0 + i\widehat{\epsilon b}_1 y| + |\psi| \\ &\leq (1 + \gamma_{3N-4})|\sum_{n=0}^{N-1} \widehat{\ell}_n z^n| + u|y\widehat{b}_1| \\ &\leq (1 + \gamma_{3N-4})\sum_{n=0}^{N-1} |\widehat{\ell}_n||z|^n + u|z||\widehat{b}_1|. \end{aligned} \tag{47}$$

Thus, through Equations (29), (30), (46) and (47), we obtain

$$\begin{aligned} |\widetilde{e} - \widehat{e}| &= |\widetilde{\epsilon b}_0 + i(\widetilde{\epsilon b}_1 y + \psi) - \widehat{\epsilon b}_0\langle 2\rangle - i(\widehat{\epsilon b}_1 y + \psi)\langle 3\rangle| \\ &\leq |(\widetilde{\epsilon b}_0 - \widehat{\epsilon b}_0) + i(\widetilde{\epsilon b}_1 - \widehat{\epsilon b}_1)y| + \gamma_3|\widehat{\epsilon b}_0 + i(\widehat{\epsilon b}_1 y + \psi)| \\ &\leq \gamma_{3N-1}\sum_{n=0}^{N-1} |\widehat{\ell}_n||z|^n + u\gamma_3|z||\widehat{b}_1|. \end{aligned} \tag{48}$$

According to Equations (36), (39) and (42), we get

$$\begin{aligned} |\widehat{\ell}_n| &\leq (1 + \gamma_5)(|\pi_n| + |\sigma_n| + |\eta_n| + |\xi_n| + |\widehat{\epsilon q}||\widehat{b}_{n+2}|) \\ &\leq \gamma_9(|a_n| + 2|z||\widehat{b}_{n+1}| + |z|^2|\widehat{b}_{n+2}|) \\ &\leq \gamma_9(1 + \gamma_{3(N-n)-4})(|a_n| + 2|z||b_{n+1}| + |z|^2|b_{n+2}|) \\ &\leq \gamma_9(1 + \gamma_{3(N-n)-4})(|a_n| + 2(N-n)|z|g_{n+1} + (N-n-1)|z|^2 g_{n+2}) \\ &\leq 3\gamma_9(1 + \gamma_{3(N-n)-4})(N-n)g_n. \end{aligned} \tag{49}$$

Then, by Equations (39), (41), (48) and (49), we obtain

$$\begin{aligned}|\widetilde{e}-\widehat{e}| \leq & 3\gamma_9\gamma_{3N-1}\sum_{n=0}^{N-1}(1+\gamma_{3(N-n)-4})(N-n)g_n + u\gamma_3(1+\gamma_{3N-4})N|z|g_1 \\ \leq & (1+\gamma_{3N-4})N(3N\gamma_9\gamma_{3N-1}+u\gamma_3)g_0 \\ \leq & 3N^2\gamma_{10}\gamma_{3N-1}g_0.\end{aligned} \tag{50}$$

Hence, when combining Equations (32), (44) and (50), we have

$$\begin{aligned}|\overline{\omega}(z)-\omega(z)| \leq & u|\omega(z)| + (1+u)3N^2(\gamma_5\gamma_{3N+1}+\gamma_{10}\gamma_{3N-1})g_0 \\ \leq & u|\omega(z)| + 3N^2\gamma_{15}\gamma_{3N+1}g_0.\end{aligned} \tag{51}$$

Thus, with $\texttt{cond}(\omega, z) = g_0/|\omega(z)|$, we deduce Equation (25). □

3.2. Running Round-Off Error Analysis

Theorem 3 gives a running error bound of Algorithm 4 to evaluate the polynomial in Equation (1) in complex floating-point arithmetic:

Theorem 3. *Assume $z, a_n \in \mathbb{F} + i\mathbb{F}$ for $n = 0, \dots, N$. Then, the running round-off error bound in the compensated Goertzel algorithm for evaluting $\omega(z) = \sum_{n=0}^{N} a_n z^n$ in floating-point arithmetic satisfies*

$$|\overline{\omega}(z) - \omega(z)| \leq (|c| \oplus \widehat{\alpha}) \oslash (1 \ominus 2u), \tag{52}$$

where

$$\begin{cases} c = \widehat{\omega}(z) \oplus \widehat{e} \ominus \overline{\omega}(z), \\ \widehat{\alpha} = \widehat{\gamma}_{3N+1} \otimes \widehat{E} \oslash (1 \ominus 6(N-1) \otimes u), \\ \widehat{E} = \widehat{Eb}_0 \oplus i\widehat{Eb}_1 \otimes |y|, \\ \widehat{Eb}_n = |\widehat{\ell}_n| \oplus |p| \otimes \widehat{Eb}_{n+1} \oplus |\widehat{q}| \otimes \widehat{Eb}_{n+2}, \\ \widehat{Eb}_0 = |\widehat{\ell}_0| \oplus |x| \otimes \widehat{Eb}_1 \oplus |\widehat{q}| \otimes \widehat{Eb}_2, \\ n = N-1, N-2, \dots, 1. \end{cases} \tag{53}$$

Proof. Assume e is the error of $\widehat{\omega}(z)$, i.e., $\widehat{\omega}(z) + e = \omega(z)$. Let $\overline{\omega}(z) + c = \widehat{\omega}(z) + \widehat{e}$. Then, we have

$$\begin{aligned}|\overline{\omega}(z) - \omega(z)| &= |\overline{\omega}(z) - (\widehat{\omega}(z) + e)| \\ &= |\overline{\omega}(z) - (\widehat{\omega}(z) + \widehat{e} - \widehat{e} + e)| \\ &\leq |c| + |e - \widehat{e}|.\end{aligned} \tag{54}$$

Considering the round-off error ϵq in Algorithm 4, from Lemma 2, we have $\epsilon q = \widehat{\epsilon q}\text{<}2\text{>}$, and thus

$$\begin{cases} \widehat{\ell}_{N-1} = \pi_{N-1}\text{<}3\text{>} + \sigma_{N-1}\text{<}3\text{>} + \eta_{N-1}\text{<}2\text{>} + \xi_{N-1}\text{<}1\text{>}, \\ \widehat{\ell}_{N-2} = \pi_{N-2}\text{<}4\text{>} + \sigma_{N-2}\text{<}4\text{>} + \eta_{N-2}\text{<}3\text{>} + \xi_{N-2}\text{<}2\text{>} - \epsilon q\widehat{b}_N\text{<}4\text{>}, \\ \quad\vdots \\ \widehat{\ell}_0 = \pi_0\text{<}4\text{>} + \sigma_0\text{<}4\text{>} + \eta_0\text{<}3\text{>} + \xi_0\text{<}2\text{>} - \epsilon q\widehat{b}_2\text{<}4\text{>}. \end{cases} \tag{55}$$

Combined with Equation (45), we can deduce that

$$\begin{aligned}|\epsilon b_0 - \widehat{\epsilon b}_0| &= |\epsilon b_0 - \text{<}3N-1\text{>}\epsilon b_0| \leq \gamma_{3N-1}Eb_0, \\ |\epsilon b_1 - \widehat{\epsilon b}_1| &= |\epsilon b_1 - \text{<}3N-4\text{>}\epsilon b_1| \leq \gamma_{3N-4}Eb_1,\end{aligned} \tag{56}$$

and

$$\begin{aligned} |\widehat{\epsilon b}_0| &= |\langle 3N-1\rangle \epsilon b_0| \leq (1+\gamma_{3N-1})Eb_0, \\ |\widehat{\epsilon b}_1| &= |\langle 3N-4\rangle \epsilon b_1| \leq (1+\gamma_{3N-4})Eb_1, \end{aligned} \tag{57}$$

where Eb_n can be derived from

$$\begin{cases} Eb_n = |\ell_n| + |p|Eb_{n+1} + |\widehat{q}|Eb_{n+2}, \\ Eb_0 = |\ell_0| + |x|Eb_1 + |\widehat{q}|Eb_2, \\ \quad \ell_n = \pi_n + \sigma_n + \eta_n + \xi_n - \epsilon q \widehat{b}_{n+2}, \\ \quad \ell_0 = \pi_0 + \sigma_0 + \eta_0 + \xi_0 - \epsilon q \widehat{b}_2, \\ \quad n = N-1, N-2, \ldots, 1. \end{cases} \tag{58}$$

Furthermore, through Equation (15), we have

$$\begin{aligned} |\widehat{\ell}_n| =& \{\{[(\pi_n+\sigma_n)\frac{1}{1+\varepsilon_1} + \eta_n]\frac{1}{1+\varepsilon_2} + \xi_n\}\frac{1}{1+\varepsilon_3} \\ & - \epsilon q \frac{1}{1+\varepsilon_4}\frac{1}{1+\varepsilon_5}\widehat{b}_{n+2}\frac{1}{1+\varepsilon_6}\}\frac{1}{1+\varepsilon_7}, \\ \widehat{Eb}_n =& [(|\widehat{\ell}_n| + |p|\widehat{Eb}_{n+1}\frac{1}{1+\varepsilon_8})\frac{1}{1+\varepsilon_9} + |\widehat{q}|\widehat{Eb}_{n+2}\frac{1}{1+\varepsilon_{10}}]\frac{1}{1+\varepsilon_{11}}, \end{aligned} \tag{59}$$

where $|\varepsilon_k| \leq u$ for $k = 1, 2, \ldots, 11$. Then, we obtain

$$|\ell_n| + |p|\widehat{Eb}_{n+1} + |\widehat{q}|\widehat{Eb}_{n+2} \leq (1+u)^6 \widehat{Eb}_n. \tag{60}$$

By induction, we get

$$Eb_0 \leq (1+u)^{6(N-1)}\widehat{Eb}_0. \tag{61}$$

Assuming that $E = Eb_0 + iEb_1|y|$, we have

$$E \leq (1+u)^{6(N-1)}(\widehat{Eb}_0 + i\widehat{Eb}_1|y|) \leq (1+u)^{6N-8}\widehat{E}. \tag{62}$$

From Equations (15)–(17) and (29), we get $\widehat{e} = \widehat{\epsilon b}_0\langle 2\rangle + i\psi\langle 2\rangle + i\widehat{\epsilon b}_1 y\langle 2\rangle = \langle 2\rangle\widetilde{e}$. Then, by Equations (56), (57) and (62), due to $\gamma_k \leq (1+u)\widehat{\gamma}_k$ and $(1+u)^n\widehat{E} \leq \widehat{E} \oslash (1 \ominus (n+1) \otimes u)$, we deduce

$$\begin{aligned} |e - \widehat{e}| &\leq |e - \widetilde{e}| + \gamma_2|\widetilde{e}| = ||\epsilon b_0 - \widehat{\epsilon b}_0| + i|\epsilon b_1 - \widehat{\epsilon b}_1||y|| + \gamma_2|\widehat{\epsilon b}_0 + i\widehat{\epsilon b}_1 y| \\ &\leq \gamma_{3N-1}(Eb_0 + iEb_1|y|) + \gamma_2(1+\gamma_{3N-1})(Eb_0 + iEb_1|y|) \\ &\leq \gamma_{3N+1}E \leq \gamma_{3N+1}(1+u)^{6N-8}\widehat{E} \leq \widehat{\gamma}_{3N+1} \otimes \widehat{E} \oslash (1 \ominus 6(N-1) \otimes u) := \widehat{\alpha}. \end{aligned} \tag{63}$$

Thus, by combining Equations (54) and (63), we obtain

$$|\overline{\omega}(z) - \omega(z)| \leq |c| + |e - \widehat{e}| \leq (|c| \oplus \widehat{\alpha}) \oslash (1 \ominus 2u). \tag{64}$$

□

Considering a dynamic error estimate to evaluate the polynomial based on Theorem 3, we can improve Algorithm 4 into Algorithm 5, whose flowchart is shown in Figure 2. We can see from Algorithm 5 that a new approximate perturbation term is obtained by combining all the dynamic error estimates in each calculation, and participating in the following computation makes the final result more accurate.

3.3. Computational Complexity

The `Horner` and its compensated algorithm `CompHorner` [12] are recalled in Algorithms 6 and 7. We shall use `TwoSum` and `TwoProd` instead of `TwoSumCplx` and

`TwoProdCplx`, while the inputs of Algorithm 7 are real floating-point numbers. A comparison of the computational costs of Algorithms 1 and 4–7 is shown in Table 2. As we can see, each of the compensated algorithms (i.e., `CompHorner` or `CompGoertzel`) can be less expensive than the others in different cases. For example, `CompGoertzel` is half as expensive as `CompHorner` when $z \in \mathbb{C}, z \neq \pm 1$ and $a_n \in \mathbb{R}$. For $z \in \mathbb{C}$, $|z| \neq 1$ and $z \neq \pm 1$, the cost of `CompGoertzel` is also less than that of `CompHorner`. Even in these cases, `CompGoertzelwErr` costs less than `CompHorner`. However, for $z \in \mathbb{R}$, `CompGoertzel` is twice as expensive as `CompHorner` regardless of a_n.

Algorithm 5 Polynomial evaluation by compensated Goertzel algorithm with a dynamic error estimate

```
Function: $[\overline{\omega}(z), \mu] = \texttt{CompGoertzelwErr}((a_n)_{n=0}^N, z)$
Require: $z = x + iy \in \mathbb{F} + i\mathbb{F}$, $(a_n)_{n=0}^N \in \mathbb{F} + i\mathbb{F}$
Ensure : $\overline{\omega}(z) \approx \sum_{n=0}^N a_n z^n$, $|\omega(z) - \overline{\omega}(z)| \leq \mu$
  $p = 2x$
  $[\widehat{q}, \widehat{\epsilon q}] = \texttt{SumOfSquares}(x, y)$
  $\widehat{b}_N = a_N, \widehat{b}_{N+1} = \widehat{\epsilon b}_N = \widehat{\epsilon b}_{N+1} = \widehat{Eb}_N = \widehat{Eb}_{N+1} = 0$
  for $n = N-1, N-2, ..., 1$
    $[r_n, \pi_n] = \texttt{TwoProdRC}(p, \widehat{b}_{n+1})$
    $[s_n, \sigma_n] = \texttt{TwoProdRC}(-\widehat{q}, \widehat{b}_{n+2})$
    $[t_n, \eta_n] = \texttt{TwoSumCplx}(r_n, s_n)$
    $[\widehat{b}_n, \xi_n] = \texttt{TwoSumCplx}(t_n, a_n)$
    $\widehat{\ell}_n = \pi_n \oplus \sigma_n \oplus \eta_n \oplus \xi_n \ominus \widehat{\epsilon q} \otimes \widehat{b}_{n+2}$
    $\widehat{\epsilon b}_n = \widehat{\ell}_n \oplus p \otimes \widehat{\epsilon b}_{n+1} \ominus \widehat{q} \otimes \widehat{\epsilon b}_{n+2}$
    $\widehat{Eb}_n = |\widehat{\ell}_n| \oplus |p| \otimes \widehat{Eb}_{n+1} \oplus |\widehat{q}| \otimes \widehat{Eb}_{n+2}$
  end
  $[r_0, \pi_0] = \texttt{TwoProdRC}(x, \widehat{b}_1)$
  $[s_0, \sigma_0] = \texttt{TwoProdRC}(-\widehat{q}, \widehat{b}_2)$
  $[t_0, \eta_0] = \texttt{TwoSumCplx}(r_0, s_0)$
  $[\widehat{b}_0, \xi_0] = \texttt{TwoSumCplx}(t_0, a_0)$
  $\widehat{\ell}_0 = \pi_0 \oplus \sigma_0 \oplus \eta_0 \oplus \xi_0 \ominus \widehat{\epsilon q} \otimes \widehat{b}_2$
  $\widehat{\epsilon b}_0 = \widehat{\ell}_0 \oplus x \otimes \widehat{\epsilon b}_1 \ominus \widehat{q} \otimes \widehat{\epsilon b}_2$
  $\widehat{Eb}_0 = |\widehat{\ell}_0| \oplus |x| \otimes \widehat{Eb}_1 \oplus |\widehat{q}| \otimes \widehat{Eb}_2$
  $[\phi, \psi] = \texttt{TwoProdRC}(y, \widehat{b}_1)$
  $\widehat{e} = \widehat{\epsilon b}_0 \oplus i(\widehat{\epsilon b}_1 \otimes y \oplus \psi)$
  $\widehat{E} = \widehat{Eb}_0 \oplus i\widehat{Eb}_1 \otimes y$
  $\widehat{\omega}(z) = \widehat{b}_0 \oplus i\phi$
  $\overline{\omega}(z) = \widehat{\omega}(z) \oplus \widehat{e}$
  $c = \widehat{\omega}(z) \oplus \widehat{e} \ominus \overline{\omega}(z)$
  $\widehat{\alpha} = (\widehat{\gamma}_{3N+1} \otimes \widehat{E}) \oslash (1 \ominus 6(N-1) \otimes u)$
  $\mu = (|c| \oplus \widehat{\alpha}) \oslash (1 \ominus 2u)$
```

Algorithm 6 Polynomial evaluation by the Horner algorithm

```
Function : $\widehat{\omega}(z) = \texttt{Horner}((a_n)_{n=0}^N, z)$
Require  : $z = x + iy \in \mathbb{F} + i\mathbb{F}$, $(a_n)_{n=0}^N \in \mathbb{F} + i\mathbb{F}$
Ensure : $\widehat{\omega}(z) \approx \sum_{n=0}^N a_n z^n$
  $\widehat{b}_{N+1} = 0$
  for $n = N, N-1, ..., 0$
    $\widehat{b}_n = \widehat{b}_{n+1} \otimes z \oplus a_n$
  end
  $\widehat{\omega}(z) = \widehat{b}_0$
```

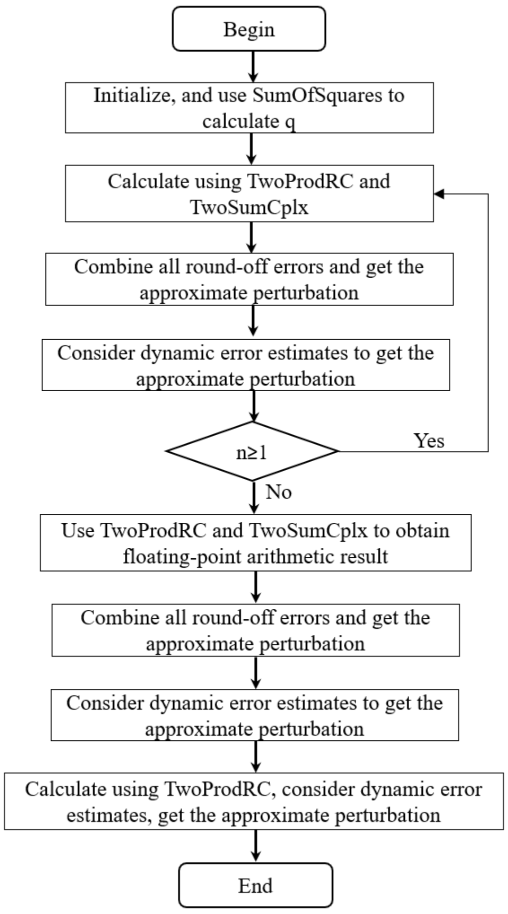
Begin
Initialize, and use SumOfSquares to calculate q
Calculate using TwoProdRC and TwoSumCplx
Combine all round-off errors and get the approximate perturbation
Consider dynamic error estimates to get the approximate perturbation
n≥1
Yes
No
Use TwoProdRC and TwoSumCplx to obtain floating-point arithmetic result
Combine all round-off errors and get the approximate perturbation
Consider dynamic error estimates to get the approximate perturbation
Calculate using TwoProdRC, consider dynamic error estimates, get the approximate perturbation
End

Figure 2. The flowchart of the compensated Goertzel algorithm with dynamic error estimates.

Algorithm 7 Polynomial evaluation by the compensated Horner algorithm

```
Function : $\overline{\omega}(z) = \texttt{CompHorner}((a_n)_{n=0}^{N}, z)$
Require : $z = x + iy \in \mathbb{F} + i\mathbb{F}$, $(a_n)_{n=0}^{N} \in \mathbb{F} + i\mathbb{F}$
Ensure : $\overline{\omega}(z) \approx \sum_{n=0}^{N} a_n z^n$
  $\widehat{b}_{N+1} = \widehat{\epsilon b}_{N+1} = 0$
  for $n = N-1, N-2..., 0$
    $[r_n, \pi_n] = \texttt{TwoProdCplx}(\widehat{b}_{n+1}, z)$
    $[\widehat{b}_n, \sigma_n] = \texttt{TwoSumCplx}(r_n, a_n)$
    $\widehat{\epsilon b}_n = \widehat{\epsilon b}_{n+1} \otimes z \oplus \pi_n \oplus \sigma_n$
  end
  $\overline{\omega}(z) = \widehat{b}_0 + \widehat{\epsilon b}_0$
```

Table 2. Comparison of computational costs of `Horner`, `Goertzel`, `CompHorner`, `CompGoertzel` and `CompGoertzelwErr` algorithms.

Variates	Coefficients	Horner	Goertzel	CompHorner	CompGoertzel	CompGoertzelwErr
$z \in \mathbb{R}$	$a_n \in \mathbb{R}$	2N	4N + 4	26N + 3	55N + 45	59N + 60
	$a_n \in \mathbb{C}$	4N	8N + 6	52N + 6	110N + 66	114N + 81
$z \in \mathbb{C}$ and $\lvert z\rvert \neq 1$	$a_n \in \mathbb{R}$	7N − 4	4N + 7	90N + 6	55N + 91	59N + 106
	$a_n \in \mathbb{C}$	8N	8N + 12	97N + 6	110N + 150	114N + 165
$z \in \mathbb{C}$, $\lvert z\rvert = 1$ and $z \neq \pm 1$	$a_n \in \mathbb{R}$	7N − 4	3N + 4	90N + 6	34N + 26	39N + 41
	$a_n \in \mathbb{C}$	8N	6N + 9	97N + 6	68N + 90	72N + 105

4. Numerical Experiments

In this section, we test the accuracy, performance and application of our algorithm. All numerical experiments are performed in IEEE-754 double precision as working precision.

4.1. Accuracy

The accuracy measurements in this part were found in MATLAB R2019b, and the exact results were obtained by the Symbolic Toolbox in order to compute the relative errors. As in [12], we considered the expanded form of the polynomial

$$\omega(z) = (z - 1 - i)^n \tag{65}$$

at $z = 1.333 + 1.333i$ for $n = 3{:}42$, while the condition number varied from 10^3 to 10^{33}. The relative accuracy of the `Horner`, `Goertzel`, `CompHorner` and `CompGoertzel` algorithms as well as the theoretical bounds of `CompGoertzel` in Theorems 2 and 3 are exhibited in Figure 3. We can observe that the `CompGoertzel` algorithm, which had almost the same accuracy as the `CompHorner` algorithm, was absolutely stable when the condition number was smaller than 10^{16}. Moreover, the numerical results and the error bounds had a good agreement, especially the running error bound, which was almost the same as the real relative errors, along with the results while the condition number was smaller than 10^{13}.

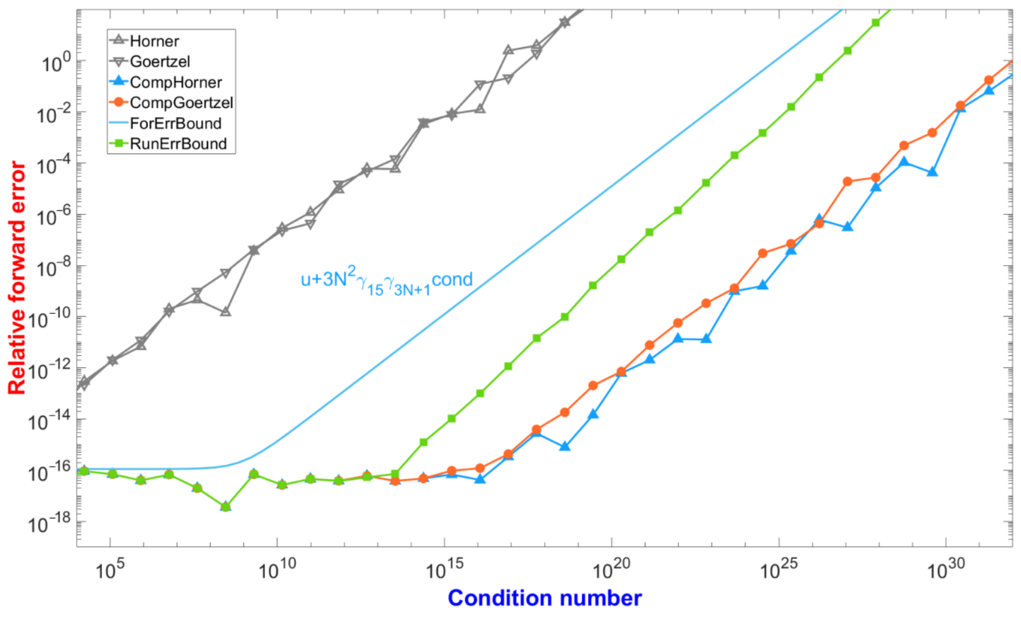

Figure 3. Accuracy of evaluation of $\omega(z) = (z - 1 - i)^n$ at $z = 1.333 + 1.333i$ for $n = 3{:}42$.

4.2. Running Time

In this part, we show the practical performance of the `Goertzel`, `CompGoertzel`, `CompHorner` and `CompGoertzelwErr` algorithms in terms of measured computing time. The tests were performed in the following environments:

- Env1: Laptop with Intel Core i7-7700 CPU, 4 cores each at 3.6 GHz and with Microsoft Visual C++ 2012 with the default compiler option /od on Windows 7;
- Env2: Node of workstation with Intel Xeon E5-2697A CPU, 16 cores each at 2.6 GHz and with gcc 7.4.0 with the default compiler option-O0 on x86_64-Ubuntu-linux 18.04.

We generated the test polynomials with random coefficients in the interval $[-1, 1]$, whose degree varied from 50 to 10,000 by a step of 50. The average time ratios for `CompGoertzel` `/Goertzel`, `CompGoertzel/CompHorner`, `CompGoertzelwErr/CompGoertzel` and `CompGo` `ertzelwErr/CompHorner` are reported in Tables 3 and 4, while the coefficients of the test polynomials were $a_n \in \mathbb{R}$ and $a_n \in \mathbb{C}$, respectively. As we can see, `CompGoertzel` was faster than `Goertzel` in practice, especially when testing in Linux. We note the good agreement of the numerical and theoretical results except for `CompGoertzelwErr`/`CompHorner` while $z \in \mathbb{C}$, $|z| \neq 1$, $z \neq \pm 1$ and $a_n \in \mathbb{C}$ in Env2. This is because `CompHorner` takes more benefit than `CompGoertzelwErr` from the Fused-Multiply-and-Add instruction [34,35] and the instruction-level parallelism [7,8]. However, `CompGoertzelwErr` was still faster than `CompHorner` in this case in Env1.

Table 3. Theoretical computational complexity and measured running time ratios in $a_n \in \mathbb{R}$.

Variates		$\frac{\text{CompGoertzel}}{\text{Goertzel}}$	$\frac{\text{CompGoertzel}}{\text{CompHorner}}$	$\frac{\text{CompGoertzelwErr}}{\text{CompGoertzel}}$	$\frac{\text{CompGoertzelwErr}}{\text{CompHorner}}$
$z \in \mathbb{R}$	Theoretical	13.75	2.12	1.07	2.27
	Evn1	9.15	1.42	1.13	1.59
	Evn2	2.76	1.35	1.11	1.49
$z \in \mathbb{C}$ and $|z| \neq 1$	Theoretical	13.75	61.14%	1.07	65.59%
	Evn1	9.29	64.19%	1.13	72.22%
	Evn2	2.81	67.71%	1.1	74.14%
$z \in \mathbb{C}$, $|z| = 1$ and $z \neq \pm 1$	Theoretical	11.33	37.79%	1.15	43.35%
	Evn1	6.53	44.09%	1.1	48.43%
	Evn2	2.22	53.66%	1.09	58.38%

Table 4. Theoretical computational complexity and measured running time ratios in $a_n \in \mathbb{C}$.

Variates		$\frac{\text{CompGoertzel}}{\text{Goertzel}}$	$\frac{\text{CompGoertzel}}{\text{CompHorner}}$	$\frac{\text{CompGoertzelwErr}}{\text{CompGoertzel}}$	$\frac{\text{CompGoertzelwErr}}{\text{CompHorner}}$
$z \in \mathbb{R}$	Theoretical	13.75	2.12	1.04	2.19
	Evn1	10.32	1.24	1.17	1.46
	Evn2	4.8	1.16	1.35	1.57
$z \in \mathbb{C}$ and $|z| \neq 1$	Theoretical	13.75	1.13	1.04	1.18
	Evn1	10.37	1.25	1.19	1.48
	Evn2	4.8	1.17	1.35	1.58
$z \in \mathbb{C}$, $|z| = 1$ and $z \neq \pm 1$	Theoretical	11.33	70.13%	1.06	74.26%
	Evn1	7.21	84.61%	1.17	98.54%
	Evn2	3.67	89.22%	1.33	1.18

4.3. Application

The test environment in this part was the same as the accuracy measurements. We considered the polynomial in Equation (1) with $a_0, a_1, \ldots, a_N \in \mathbb{R}$ and $z_k = e^{-2\pi k i/(N+1)}$. Then, the DFT, which could be computed by the function "fft" with the polynomial's coefficients in MATLAB returned $\omega(z_k)$ for $k = 0, 1, \ldots, N$. The `Goertzel` algorithm can also compute the DFT with the polynomial's coefficients of specific indices z_k in a vector. Figure 4 shows the relative errors of `Goertzel`, `CompGoertzel`, and `fft` applied to polynomials whose degrees varied from 50 to 1000 by a step of 10 with random coefficients in the interval $[-1, 1]$, where the relative error was defined as $\|\text{res}_{\text{comput}} - \text{res}_{\text{exact}}\|_2 / \|\text{res}_{\text{exact}}\|_2$. In Figure 4, although `fft` was more accurate than `Goertzel`, the relative errors of `Goertzel` and `fft` were all increasing, while the degree of the polynomial grew. However, `CompGoertzel` always obtained full-precision accurate results in this test, and the relative error of our algorithm was 10^{15} to 10^{17}, while the `fft` was from 10^{12} to 10^{15}.

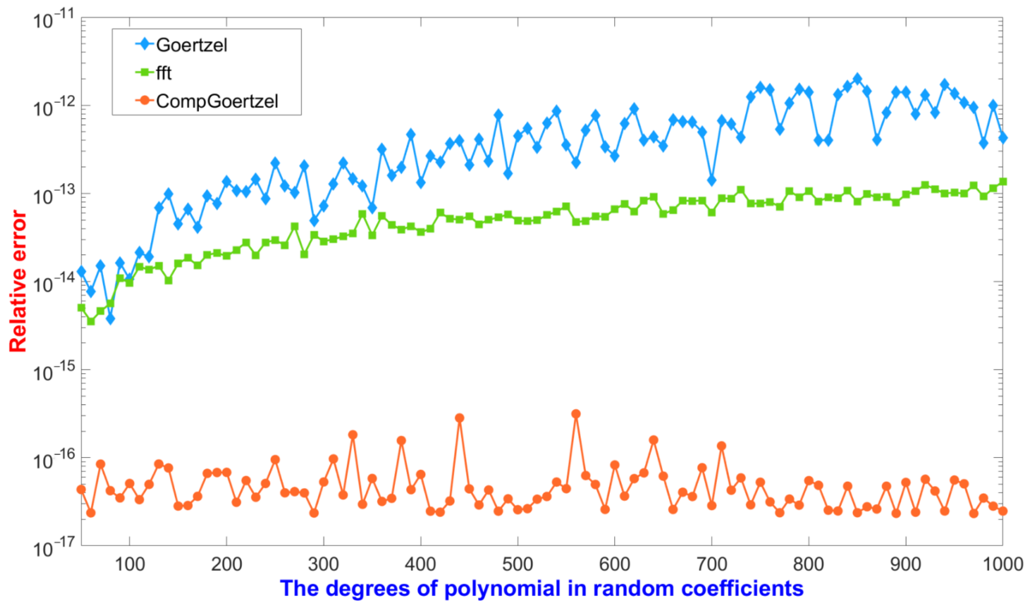

Figure 4. The relative errors of DFT for polynomials with random coefficients.

5. Conclusions

In this paper, we presented a compensated Goertzel algorithm with dynamic error estimation to evaluate polynomials in complex floating-point arithmetic. The forward error analysis and numerical experiments show that the algorithm can yield full working precision accuracy. Furthermore, although the algorithm is as precise as the compensated Horner algorithm, it is quicker in certain situations. The algorithm also performed well in the application of computing the DFT of specific indices.

Author Contributions: Conceptualization, C.L., P.D. and K.L.; methodology, C.L., P.D., Y.L. and H.J.; validation, C.L., K.L. and Z.Q.; formal analysis, K.L., Y.L. and H.J.; investigation, C.L., P.D. and H.J.; resources, Y.L. and Z.Q.; data curation, C.L. and P.D.; writing—original draft preparation, C.L., P.D., K.L. and Y.L.; writing—review and editing, C.L., H.J. and Z.Q.; supervision, H.J. and Z.Q.; project administration, P.D. and H.J.; funding acquisition, P.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Natural Science Foundation of China (No. 61907034), the 173 program of China (2020-JCJQ-ZD-029), the National Key Research and Development Program of China (2020YFA0709803), the Science Challenge Project of China (TZ2016002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank the reviewers for providing valuable comments about our article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DFT	discrete Fourier transform
VS	Volk and Schumaker

References

1. Peña, J.M.; Sauer, T. On the multivariate Horner scheme. *Siam J. Numer. Anal.* **2000**, *37*, 1186–1197. [CrossRef]
2. Gentleman, W.M. An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients. *Comput. J.* **1969**, *12*, 160–165. [CrossRef]
3. Newbery, A.C.R. Error analysis for Fourier series evaluation. *Math. Comput.* **1973**, *27*, 639–644. [CrossRef]
4. Wilkinson, J.H. *Rounding Errors in Algebraic Processes*; Courier Corporation: Englewood Cliffs, NJ, USA, 1994.
5. Smoktunowicz, A.; Wróbel, I. On improving the accuracy of Horner's and Goertzel's algorithms. *Numer. Algorithms* **2005**, *38*, 243–258. [CrossRef]
6. Bailey, D.H. Library for Double-Double and Quad-Double Arithmetic. Available online: http://www.nersc.gov/dhbailey/mpdist/mpdist.html (accessed on 18 February 2021).
7. Louvet, N. *Compensated Algorithms in Floating-Point Arithmetic: Accuracy, Validation, Performances*; Université de Perpignan Via Domitia: Perpignan, France, 2007.
8. Langlois, P.; Louvet, N. *More Instruction Level Parallelism Explains the Actual Efficiency of Compensated Algorithm*; Technical Report hal-00165020; DALI Research Team, University of Perpignan: Perpignan, France, 2007.
9. Ogita, T.; Rump, S.M.; Oishi, S. Accurate sum and dot product. *Siam J. Sci. Comput.* **2005**, *26*, 1955–1988. [CrossRef]
10. Graillat, S.; Langlois, P.; Louvet, N. Algorithms for accurate, validated and fast polynomial evaluation. *Jpn. J. Ind. Appl. Math.* **2009**, *26*, 191–214. [CrossRef]
11. Graillat, S.; Morain, V. Error-free transformations in real and complex floating-point arithmetic. In Proceedings of the International Symposium on Nonlinear Theory and Its Applications, Vancouver, BC, Canada, 16–19 September 2007 ; pp. 341–344.
12. Graillat, S.; Morain, V. Accurate summation, dot product and polynomial evaluation in complex floating-point arithmetic. *Inf. Comput.* **2012**, *216*, 57–71. [CrossRef]
13. Graillat, S. An accurate algorithm for evaluating rational functions. *Appl. Math. Comput.* **2018**, *337*, 494–503. [CrossRef]
14. Cameron, T.; Graillat, S. On a Compensated Ehrlich-Aberth Method for the Accurate Computation of All Polynomial Roots. Available online: https://hal.archives-ouvertes.fr/hal-03335604 (accessed on 16 March 2021).
15. Jiang, H.; Barrio, R.; Li, H.; Liao, X.; Cheng, L.; Su, F. Accurate evaluation of a polynomial in Chebyshev form. *Appl. Math. Comput.* **2011**,*217*, 9702–9716. [CrossRef]
16. Jiang, H.; Li, S.; Cheng, L.; Su, F. Accurate evaluation of a polynomial and its derivative in Bernstein form. *Comput. Math. Appl.* **2010**, *60*, 744–755. [CrossRef]
17. Delgado, J.; Peña, J.M. Algorithm 960: POLYNOMIAL: An Object-Oriented Matlab Library of Fast and Efficient Algorithms for Polynomials. *ACM Trans. Math. Softw.* **2016**, *42*, 1–19. [CrossRef]
18. Kazal, N.Y.; Mukhlash, I.; Sanjoyo, B.A.; Hidayat, N.; Ozaki, K. Extended use of error-free transformation for real matrix multiplication to complex matrix multiplication. *Siam J. Phys. Conf. Ser.* **2021**, *1821*, 012022. [CrossRef]
19. Ozaki, K. Error-free transformation of matrix multiplication for multi-precision computations. In Proceedings of the 19th International Symposium on Scientific Computing, Computer Arithmetic, and Verified Numerical Computations, Szeged, Hungary, 13–15 September 2021 ; Volume 33.
20. Ozaki, K. An Error-Free Transformation for Matrix Multiplication with Reproducible Algorithms and Divide and Conquer Methods. *J. Phys. Conf. Ser.* **2020**, *1490*, 012062. [CrossRef]
21. Ozaki, K.; Ogita, T. The Essentials of verified numerical computations, rounding error analyses, interval arithmetic, and error-free transformations. *Nonlinear Theory Its Appl.* **2020**, *11*, 279–302. [CrossRef]
22. Blanchard, P.; Higham, D.J.; Higham, N.J. Accurately computing the log-sum-exp and softmax functions. *IMA J. Numer. Anal.* **2021**, *41*, 2311–2330. [CrossRef]
23. Delgado, J.; Peña, J.M. Running relative error for the evaluation of polynomials. *SIAM J. Sci. Comput.* **2009**, *31*, 3905–3921. [CrossRef]
24. Jiang, H.; Graillat, S.; Barrio, R.; Yang, C. Accurate, validated and fast evaluation of elementary symmetric functions and its application. *Appl. Math. Comput.* **2016**, *273*, 1160–1178. [CrossRef]
25. Barrio, R.; Du, P.; Jiang, H.; Serrano, S. ORTHOPOLY: A library for accurate evaluation of series of classical orthogonal polynomials and their derivatives. *Comput. Phys. Commun.* **2018**, *231*, 146–162. [CrossRef]
26. Croci, M.; Fasi, M.; Higham, N.J.; Mary, T.; Mikaitis, M. Stochastic rounding: Implementation, error analysis and applications. *R. Soc. Open Sci.* **2022**, *9*, 211631. [CrossRef]
27. *IEEE Standard 754-2008*; Standard for Binary Floating Point Arithmetic. ANSI: New York, NY, USA, 2008.
28. Clenshaw, C.W. A note on the summation of Chebyshev series. *Math. Comput.* **1955**, *9*, 118–120. [CrossRef]
29. Szegö, G. *Orthogonal Polynomials*; American Mathematical Society: Providence, RI, USA, 1939.
30. Knuth, D.E. *The Art of Computer Programming: Seminumerical Algorithms*, 3rd ed.; Addison-Wesley: Boston, MA, USA, 1998.
31. Dekker, T.J. A floating-point technique for extending the available precision. *Numer. Math.* **1971**, *18*, 224–242. [CrossRef]
32. Graillat, S.; Lauter, C.; Tang, P.T.; Yamanaka, N.; Oishi, S. Efficient Calculations of Faithfully Rounded l_2-Norms of n-Vectors. *ACM Trans. Math. Softw.* **2015**, *41*, 1–20. [CrossRef]
33. Higham, N.J. *Accuracy and Stability of Numerical Algorithms*, 2nd ed.; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 2002.

34. Markstein, P. *IA-64 and Elementary Functions: Speed and Precision*; Prentice-Hall: Englewood Cliffs, NJ, USA, 2000.
35. Nievergelt, Y. Scalar fused multiply-add instructions produce floating-point matrix arithmetic provably accurate to the penultimate digit. *ACM Trans. Math. Softw.* **2003**, *29*, 27–48. [CrossRef]