On improving the accuracy of Horner's and Goertzel's algorithms

Alicja Smoktunowicz and Iwona Wróbel

Faculty of Mathematics and Information Science, Warsaw University of Technology, Pl.Politechniki 1, 00-661 Warsaw, Poland

e-mail: smok@mini.pw.edu.pl, wrubelki@wp.pl

Abstract

It is known that Goertzel's algorithm is much less numerically accurate than the Fast Fourier Transform (FFT)(Cf. [2]). In order to improve accuracy we propose modifications of both Goertzel's and Horner's algorithms based on the divide-and-conquer techniques. The proof of the numerical stability of these two modified algorithms is given. The numerical tests in Matlab demonstrate the computational advantages of the proposed modifications. The appendix contains the proof of numerical stability of Goertzel's algorithm of polynomial evaluation.

AMS subject classification: 65F35, 65G50.

Keywords: Numerical stability, polynomial evaluation, FFT.

1 Introduction

The aim of this paper is to improve the accuracy of polynomial evaluation, mainly Horner's and Goertzel's algorithms. Both, Horner's and Goertzel's methods are frequently used in the interpolation and approximation problems and in signal processing. Goertzel's algorithm is implemented in Matlab, it's included in the Signal Processing Toolbox. The function "fft" returns the Discrete Fourier Transform (DFT) computed with a Fast Fourier

Transform (FFT) algorithm and the function "goertzel" computes DFT of specific indices in a vector.

In this paper we consider more general case of evaluating a polynomial

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

where $z \in \mathbb{C}$ and $a_0, \ldots, a_N \in \mathbb{C}$.

Note that for $z = e^{i\xi}$ and $a_0, \ldots, a_N, \xi \in \mathbb{R}$ we have

$$w(e^{i\xi}) = \sum_{n=0}^{N} a_n \cos n\xi + i \sum_{n=0}^{N} a_n \sin n\xi.$$

DFT returns $y_k = w(z_k)$, $k = 0, \ldots, N$, where $z_k = e^{i\xi_k} = e^{-\frac{2\pi i k}{N+1}}$ (Cf. [6], p. 10).

It is observed (see "help goertzel" in Matlab Signal Processing Toolbox) that compared with the Fast Fourier Transform algorithm (FFT), Goertzel's algorithm is much less numerically accurate, which can be visible especially for high-scale problems.

We propose the algorithm PEMA (Polynomial Evaluation Modified Algorithm), which is based on the repetitive use of some algorithm W for evaluating polynomials. This algorithm can be e.g. Horner's or Goertzel's scheme. The cost of PEMA is comparable to the cost of W and the error bound of PEMA may be significantly smaller than the error bound of W. We prove that if W is stable then PEMA is also numerically stable (see section 3.2). In practice, one should use only numerically stable algorithms.

We say that an algorithm of evaluating (1) is componentwise backward stable with respect to the data $a_0, \ldots, a_N \in \mathbb{C}$ and $z \in \mathbb{C}$ if the value $\tilde{w}(z)$ computed by this algorithm is an exact value of a polynomial for slightly perturbed coefficients a_n and z, i.e.

$$\tilde{w}(z) = \sum_{n=0}^{N} [a_n(1+\mu_n)]\,[z(1+\beta)]^n, \quad |\mu_n| \le A_N \epsilon_M, \quad |\beta| \le Z_N \epsilon_M, \tag{2}$$

where A_N and Z_N are modestly growing functions of N and ϵ_M is the machine precision.

Throughout the paper we assume that the coefficients of a polynomial $w(z)$ are complex.

In the error analysis of PEMA we consider perturbations not only of polynomial coefficients, but also of z. Notice that usually the exact value of z is not known, e.g. z is given as $z = e^{i\xi}$. Then $z = c + i\,s$, $c = \cos\xi$, $s = \sin\xi$ and $\tilde{c} = c + \Delta c$, $\tilde{s} = s + \Delta s$, $|\Delta c|$, $|\Delta s| \le \nu\,\epsilon_M$, where ν is small. Then the perturbed value $\tilde{z}$ can be written as $\tilde{z} = z\,(1+\eta)$, $|\eta| \le \sqrt{2}\,\nu\,\epsilon_M$.

Then with help of Taylor expansion (2) leads to

$$|\,\tilde{w}(z) - w(z)\,| \le |\,\beta\,|\,|\,z\,w'(z)\,| + \sum_{n=0}^{N} |\,a_n\,|\,|\,\delta_n\,|\,|\,z\,|^n + \mathcal{O}(\epsilon_M^2),$$

and further

$$|\,\tilde{w}(z) - w(z)\,| \le \epsilon_M (A_N \sum_{n=0}^{N} |\,a_n\,|\,|\,z\,|^n + Z_N\,|\,z\,|\,|\,w'(z)\,|) + \mathcal{O}(\epsilon_M^2).$$

Numerical stability of Horner's algorithm was first given by Wilkinson (Cf. [13], pp. 36-37, 49-50) who proved that $Z_N = 0$ and $A_N \approx 2N$, provided that the data $a_0, \ldots, a_N \in \mathbb{R}$ and $z \in \mathbb{R}$ are exactly representable in floating point arithmetic (fl). Despite of a bad reputation of Goertzel's algorithm as a method of computing Fourier series $\sum_{n=0}^{N} a_n \cos n\xi$ and $\sum_{n=1}^{N} a_n \sin n\xi$ with respect to the data $a_0, \ldots, a_N \in \mathbb{R}$ and a given argument $\xi \in \mathbb{R}$ (Cf. [11], pp. 84-88, [2], [7], [8]) we prove that Goertzel's algorithm is numerically stable in a sense (2). The respective constants are $Z_N = 0$ and A_N is of order N^2, provided that the data $a_0, \ldots, a_N$ and z are exactly representable in fl (see Theorem 2 and Table 0).

In order to improve accuracy we propose modifications of both Goertzel's and Horner's algorithms based on the divide-and-conquer techniques. The idea is not quite new, there are numerous divide-and-conquer parallel algorithms for polynomial evaluation (Cf. [3], p. 70). The goal of our work is to split a polynomial in "the proper way" in order to refine results. We show that the constants A_N and Z_N in (2) can be significantly decreased, in comparison with the classical Horner's and Goertzel's algorithms, which is of great importance for large N (see Table 0 in section 3), e.g. for $N = 2^p$

our divide-and-conquer algorithm PEMA results in A_N of order $log_2 N$ and Z_N of order unity.

Tests included in section 4 confirm theoretical results. We also implemented Reinsch's modification of Goertzel's algorithm (Cf. [11], pp. 86-88) for evaluation of (1), but it turned out that the numerical results were comparable to these given by standard Goertzel's algorithm. For this reason we don't include them in section 4 devoted to numerical experiments.

2 Classical polynomial evaluation schemes

The Horner scheme is the standard method for evaluation of a polynomial (1) at a given point $z \in \mathbb{C}$. We assume that $a_0, \ldots, a_N \in \mathbb{C}$. We write $w(z)$ as follows

$$w(z) = a_0 + z(a_1 + z(\ldots + z(a_{N-1} + za_N)\ldots)).$$

Algorithm 1 (Horner's rule)

$w := 0$
for $\quad n = N, N-1, \ldots, 0$
$\qquad w := a_n + z\,w$
end
$w(z) := w$

The complexity of Horner's algorithm $C_N(H)$, counted as a number of multiplications is equal to N, which gives in general $C_N(H) = 4N$ real multiplications. We assume that the product of two complex numbers is computed in a natural way and in consequence one complex multiplication is equivalent to four real ones.

The idea of Goertzel's algorithm is different. Suppose $z = x + iy$. Divide a polynomial $w(\lambda) = \sum_{n=0}^{N} a_n \lambda^n$ by a quadratic polynomial $(\lambda - z)(\lambda - \overline{z}) = \lambda^2 - \hat{p}\lambda - \hat{q}$ with real coefficients $\hat{p}$ and $\hat{q}$, where $\hat{p} = 2x$ and $\hat{q} = -|z|^2$.

Then

$$w(\lambda) = (\lambda - z)(\lambda - \overline{z}) \sum_{n=2}^{N} b_n \lambda^{n-2} + b_0 + b_1 \lambda$$

and, consequently, $w(z) = b_0 + b_1 z$. This leads to the following

Algorithm 2 (Goertzel's algorithm)

$\hat{p} := 2x$

$\hat{q} := -(x^2 + y^2)$

$b_{N+1} := 0$

$b_N := a_N$

for $n = N-1, \dots, 1$

    $b_n := a_n + \hat{p}\; b_{n+1} + \hat{q}\; b_{n+2}$

end

$u := (a_0 + x\, b_1 + \hat{q}\, b_2)$

$v := y\, b_1$

$w(z) := u + i\, v$

In general, the number of real multiplications needed by Goertzel's method is the same as those needed by Horner's algorithm. However, in special cases each of these algorithms can be less expensive than the other. For example, for $z \in \mathbb{R}$ Goertzel's algorithm is twice as expensive as Horner's rule regardless of the polynomial coefficients. On the other hand consider the case of polynomial with real coefficients and $z \in \mathbb{C}$, $|z| = 1$. Then all b_n are real, $\hat{q} = 1$ and $b_n = a_n + \hat{p}\; b_{n+1} + \; b_{n+2}$ for $n = N-1, \dots, 1$. The complexity of Goertzel's method reduces to N while the cost of Horner's rule is still $4N$.

Note that if $z = 1$, then $w(z) = \sum_{n=0}^{N} a_n$ and Horner's rule is nothing else but a backward summation.

We now derive an algorithm based on the divide-and-conquer technique.

3 A new polynomial evaluation modified algorithm (PEMA)

Suppose a polynomial $w(z)$ is given by $w(z) = \sum\limits_{n=0}^{N} a_n z^n$ where $N = s^p$ and $s > 1$. We can write $w(z)$ in the following form:

$$w(z) = \{a_0 + a_1 z + \cdots + a_{s-1} z^{s-1}\} + \{a_s + a_{s+1} z + \cdots + a_{2s-1} z^{s-1}\} z^s + \cdots +$$
$$+ \{a_{(s^{p-1}-1)s} + a_{(s^{p-1}-1)s+1} z + \cdots + a_{s^p-1} z^{s-1}\} (z^s)^{s^{p-1}-1} + a_{s^p} (z^s)^{s^{p-1}} =$$
$$= a_0^{(1)} + a_1^{(1)} z_1 + a_2^{(1)} z_1^2 + \cdots + a_{s^{p-1}}^{(1)} z_1^{s^{p-1}} = \sum_{j=0}^{s^{p-1}} a_j^{(1)} z_1^j,$$

where $z_1 = z^s$, $a_j^{(1)} = \sum_{k=0}^{s-1} a_{js+k}^{(0)} z^k$, $j = 0, 1, \ldots, s^{p-1} - 1$, $a_{s^{p-1}}^{(1)} = a_{s^p}^{(0)}$, and $a_j^{(0)} = a_j$ for $j = 0, 1, \ldots, N$.

Now we can interpret $\sum_{j=0}^{s^{p-1}} a_j^{(1)} z_1^j$ as a polynomial of variable z_1 with the coefficients $a_j^{(1)}$ and proceed in the same manner as before. We continue this process and for $m = 0, 1, \ldots, p-1$ write $w(z)$ as follows

$$w(z) = \sum_{j=0}^{s^{p-m}} a_j^{(m)} z_m^j,$$

where $z_0 = z$ and $z_m = z_{m-1}^s$ for $m = 1, 2, \ldots, p-1$.

It is easy to prove that for $m = 1, 2, \ldots, p-1$ and $j = 0, 1, \ldots, s^{p-m} - 1$

$$a_j^{(m)} = \sum_{r=0}^{s^m-1} a_{js^m+r}\, z^r. \tag{3}$$

For complexity and computational accuracy reasons we don't evaluate (3) directly, by Horner or Goertzel algorithm for polynomial of variable z and degree $s^m - 1$, but use the relation

$$a_j^{(m)} = \sum_{k=0}^{s-1} a_{js+k}^{(m-1)} z_{m-1}^k.$$

Notice that $a_j^{(m)}$ is a polynomial of variable z_{m-1} and degree $s-1$.

More precisely, given an algorithm W for evaluating polynomials, e.g. Horner's or Goertzel's algorithm, we produce a new divide-and-conquer algorithm.

Algorithm 3 (PEMA)

This algorithm uses the divide-and-conquer method to compute $w(z)$ where $z \in \mathbb{R}$ or $z \in \mathbb{C}$. The coefficients a_n may be either complex or real.

1. $z_0 = z$
   $a_j^{(0)} = a_j \quad \text{for} \quad j = 0, 1, \ldots, N$

2. for $\quad m = 1, \ldots, p-1$

   $z_m = z_{m-1}^s$
   $a_{s^{p-m}}^{(m)} = a_{s^{p-(m-1)}}^{(m-1)} = a_N$
   for $\quad j = 0, 1, \ldots, s^{p-m} - 1$
   compute $\quad a_j^{(m)} = \sum\limits_{k=0}^{s-1} a_{js+k}^{(m-1)} z_{m-1}^k \quad$ by algorithm W
   end

   end

3. compute $\quad w(z) = \sum\limits_{j=0}^{s} a_j^{(p-1)} z_{p-1}^j \quad$ by algorithm W

Note that $p = 1$ implies $N = s$ and PEMA is nothing else but W applied to $w(z)$.

PEMA is an extension of a summation algorithm proposed in [4]. For $N = 2^p$ and $z = 1$ PEMA coincides with the log-sum algorithm.

3.1 Total cost of PEMA

Suppose the complexity of the algorithm W is $C_N = bN$, $b = const$, i.e. W needs C_N multiplications to compute $w(z) = \sum\limits_{n=0}^{N} a_n z^n$. We give a formula for complexity of PEMA valid under assumption that z_m is computed in a natural way, (see section 3.2):

$$C(PEMA)=\sum_{m=1}^{p-1}\{\sum_{j=0}^{s^{p-m}-1}C_{s-1}+(s-1)\}+C_s=(s-1)(p-1)+C_s+C_{s-1}\frac{s(s^{p-1}-1)}{s-1}.$$

According to this formula the complexity of PEMA with Horner is equal to $C_N+(s-1)(p-1)$. Very often the latter term is not significant in comparison with C_N.

Remark. Each $a_j^{(m)}$, $j=0,1,\ldots,s^{p-m}-1$ can be computed independently. It's a big advantage of PEMA because of possibility of parallel implementation, which can be useful especially for really large problems.

3.2 Error analysis of PEMA

We consider complex arithmetic (cfl) implemented using standard real arithmetic with machine precision ϵ_M. Then

$$(4)\qquad cfl(x+y)=(x+y)(1+\delta),\qquad |\,\delta\,|\le\epsilon_M \qquad \text{for } x,y\in\mathbb{C}$$

and provided that the product xy is computed using an ordinary algorithm we have (Cf. [5])

$$(5)\qquad cfl(xy)=(xy)(1+\eta),\qquad |\,\eta\,|\le c\,\epsilon_M,$$

where

$$(6)\qquad c=\begin{cases}1 & \text{for}\quad x,y\in\mathbb{R}\quad\text{or}\quad x\in\mathbb{R},\ y\in\mathbb{C}\\ 1+\sqrt{2} & \text{for}\quad x,y\in\mathbb{C}.\end{cases}$$

The value $z_m=z_{m-1}^s$ is determined in a natural way by computing the consecutive powers of z_m, i.e. z_{m-1}, z_{m-1}^2, $\ldots$, z_{m-1}^s.
Then

$$(7)\qquad \tilde{z}_m=cfl(\tilde{z}_{m-1}^s)=\tilde{z}_{m-1}^s(1+\delta_m),\qquad |\,\delta_m\,|\le(s-1)\,c\,\epsilon_M+\mathcal{O}(\epsilon_M^2).$$

Now we are in a position to give the error analysis of the PEMA algorithm. For simplicity we assume that $a_0,\ldots,a_N$ and z are represented exactly in cfl and that s and p are fixed, $N = s^p$. We also assume that the result given by the algorithm W of evaluating $w(z) = \sum_{n=0}^{N} a_n z^n$ in cfl satisfies

$$\tilde{w}(z) = \sum_{n=0}^{N} a_n (1 + \Delta_n)\, z^n, \qquad |\, \Delta_n \,| \leq A_N\, \epsilon_M, \tag{8}$$

where A_N is an increasing function of N. W in PEMA can be Horner's or Goertzel's rule. For detailed information on A_N see (37).

For $m = 1, \ldots, p-1$ and $j = 0, 1, \ldots, s^{p-m} - 1$ the values $\tilde{a}_j^{(m)}$, computed in cfl, can be written as follows

$$\tilde{a}_j^{(m)} = \sum_{k=0}^{s-1} \tilde{a}_{js+k}^{(m-1)} (1 + \Delta_{j,k}^{(m)})\, \tilde{z}_{m-1}^{k}, \qquad |\, \Delta_{j,k}^{(m)} \,| \leq A_{s-1}\, \epsilon_M. \tag{9}$$

The formula (7) allows us to write $\tilde{z}_{m-1}$ in the following way

$$\tilde{z}_{m-1} = [z(1 + \gamma_m)]^{s^{m-1}}, \qquad 1 + \gamma_m = \prod_{t=1}^{m-1} (1 + \delta_t)^{\frac{1}{s^t}}. \tag{10}$$

From (7) we obtain an upper bound for $|\, \gamma_m \,|$

$$|\, \gamma_m \,| \leq (s-1)\, c\, \epsilon_M \sum_{t=1}^{m-1} \frac{1}{s^t} + \mathcal{O}(\epsilon_M^2).$$

Thus

$$|\, \gamma_m \,| \leq c\, \epsilon_M + \mathcal{O}(\epsilon_M^2). \tag{11}$$

Lemma 1 *Assume that* $cfl(z) = z$ *and* $cfl(a_n) = a_n$, $n = 0, \ldots, N$ *and* $N = s^p$. *Suppose that* $A_N\, \epsilon_M \leq 0.1$ *and that* (7-9) *hold. Then for* $m = 1, \ldots, p-1$ *and* $j = 0, 1, \ldots, s^{p-m} - 1$

$$\tilde{a}_j^{(m)} = \sum_{r=0}^{s^m - 1} [a_{js^m + r}(1 + \eta_{j,r}^{(m)})]\, [z(1 + \gamma_m)]^r \tag{12}$$

where

$$|\, \gamma_m \,| \leq c\, \epsilon_M + \mathcal{O}(\epsilon_M^2), \tag{13}$$

and

$$|\, \eta_{j,r}^{(m)} \,| \leq m\, d\, \epsilon_M + \mathcal{O}(\epsilon_M^2), \qquad d = A_{s-1} + (s-1)\, c, \tag{14}$$

where c *is defined by* (6).

Proof. Let $m = 1$. Then from (9) it follows that

$$\tilde{a}_j^{(1)} = \sum_{k=0}^{s-1} a_{js+k}(1 + \Delta_{j,k}^{(1)})\, z^k, \qquad |\, \Delta_{j,k}^{(1)} \,| \leq A_{s-1}\, \epsilon_M, \tag{15}$$

which can be rewritten in the following form

$$\tilde{a}_j^{(1)} = \sum_{k=0}^{s-1} a_{js+k} \frac{(1 + \Delta_{j,k}^{(1)})}{(1+\gamma_1)^k}\, [z(1+\gamma_1)]^k = \sum_{k=0}^{s-1} [a_{js+k}(1 + \eta_{j,k}^{(1)})\, [z(1+\gamma_1)]^k,$$

where

$$1 + \eta_{j,k}^{(1)} = \frac{1 + \Delta_{j,k}^{(1)}}{(1+\gamma_1)^k}.$$

From this we obtain

$$|\, \eta_{j,k}^{(1)} \,| \leq |\, \Delta_{j,k}^{(1)} \,| + k \,|\, \gamma_1 \,| + \mathcal{O}(\epsilon_M^2).$$

Now using (11), (15), the definition of d in (14) and the fact that $k \leq s-1$ we have

$$|\, \eta_{j,k}^{(1)} \,| \leq (A_{s-1} + (s-1)\, c)\, \epsilon_M + \mathcal{O}(\epsilon_M^2) = d\, \epsilon_M + \mathcal{O}(\epsilon_M^2).$$

In the same manner, using the equality

$$[z(1+\gamma_m)] = [z(1+\gamma_{m-1})]\, (1 + \delta_{m-1})^{\frac{1}{s^{m-1}}}$$

we get $|\, \eta_{j,k}^{(m)} \,| \leq m\, d\, \epsilon_M + \mathcal{O}(\epsilon_M^2)$, which is the desired conclusion. ■

Theorem 1 *Under the assumptions of Lemma 1 the value $\tilde{w}(z)$ computed by PEMA satisfies*

$$\tilde{w}(z) = \sum_{n=0}^{N} [a_n(1+\Delta_n)]\, [z(1+\beta)]^n,$$

where

$$|\, \beta \,| = |\, \gamma_p \,| \leq c\, \epsilon_M + \mathcal{O}(\epsilon_M^2), \qquad |\, \Delta_n \,| \leq p\, (A_s + s\, c)\, \epsilon_M + \mathcal{O}(\epsilon_M^2).$$

Proof. Let $m = p-1$. Lemma 1 yields

$$\tilde{a}_j^{(p-1)} = \sum_{r=0}^{s^{p-1}-1} [a_{js^{p-1}+r}(1+\eta_{j,r}^{(p-1)})]\,[z(1+\gamma_{p-1})]^r,$$

where

$$|\,\eta_{j,r}^{(p-1)}\,| \leq (p-1)\,d\,\epsilon_M + \mathcal{O}(\epsilon_M^2).$$

By assumptions, we have

$$\tilde{w}(z) = \sum_{j=0}^{s} \tilde{a}_j^{(p-1)}(1+\xi_j^{(p)})\,\tilde{z}_{p-1}^j, \qquad |\,\xi_j^{(p)}\,| \leq A_s\,\epsilon_M.$$

This gives immediately the assertion of the theorem. ■

So, if the algorithm W satisfies (8), PEMA is numerically stable in a sense (2).

Table 0: Constants A_N and Z_N for all algorithms.

Algorithm	A_N	Z_N
Horner	$(c+1)N$	0
Goertzel	$10N^2$	0
PEMA(Horner)	$ps(2c+1)$	c
PEMA(Goertzel)	$10ps^2 + psc$	c

N is the degree of the polynomial, $c = 1$ for real coefficients $a_0, \ldots, a_N$ and $c = 1 + \sqrt{2}$ for complex a_n. Here p and s are the parameters of PEMA, $N = s^p$. Note that for $N = 2^p$, the partial polynomials in PEMA are of degree 1 and A_N is of order $\log_2 N$, which is a significant improvement when compared with the standard versions of both algorithms.

4 Numerical tests

This paragraph contains the results of the tests performed in Matlab, version 6.1.0450 (R12.1) with machine precision $\epsilon_M \approx 2.2 \cdot 10^{-16}$. We implemented all methods and compared the results they gave. Of course, it would be the most natural to compare the result given by each of the methods with the exact one. However, there are obvious obstructions, i.e. for fractional polynomial coefficients or the point z there is no way to obtain the exact value of $w(z)$. To deal with these difficulties we used the Matlab function "fft", which is perfectly stable (for details see [6], pp. 22-45). The function $yfft = fft(a)$ computes Fourier coefficients, namely $y_k = w(z_k)$, $k = 0, \ldots, N$, where $w(z)$ is the polynomial (1) and $z_k = \omega^k$, $\omega = e^{-\frac{2\pi i}{N+1}}$ is the $(N+1)$st root of unity: $\omega^{N+1} = 1$. The values z_k were computed by the Direct Call algorithm, i.e. $z_k := \cos(kt) - i\sin(kt)$, $t = \frac{2\pi}{N+1}$, which is known to be very accurate (Cf. [6], pp. 23-24).

We computed the relative error

$$error = \frac{\|y - yfft\|_2}{\|yfft\|_2} \tag{16}$$

where y denotes the vector of results given by Horner's, Goertzel's or PEMA algorithm for a certain set of points $\{z_j\}$ and $yfft$ is the result given by the function "fft" for the same set of points, namely for z_j, where $j \in \{0, 1, 9, 99, 199, 256, 299, 399, 499, 699\}$. The parameter p in PEMA (see section 3) was equal to 2, namely $N = s^2$ (i.e. $s = \sqrt{N}$).

The function "fft" can be used provided that $|z| \leq 1$. In general this condition is not needed, all algorithms, namely Goertzel's, Horner's and both versions of PEMA work for any $z \in \mathbb{C}$.

Figure 1 describes the results for Goertzel's algorithm and PEMA with Goertzel's method applied to polynomials with random coefficients.

Both graphs illustrate the logarithm of *error* (16) plotted against the logarithm of the polynomial degree n, which varies between 2^{10} and 2^{22}. The lower graph represents results given by PEMA, while the upper one these given by the standard Goertzel's algorithm.

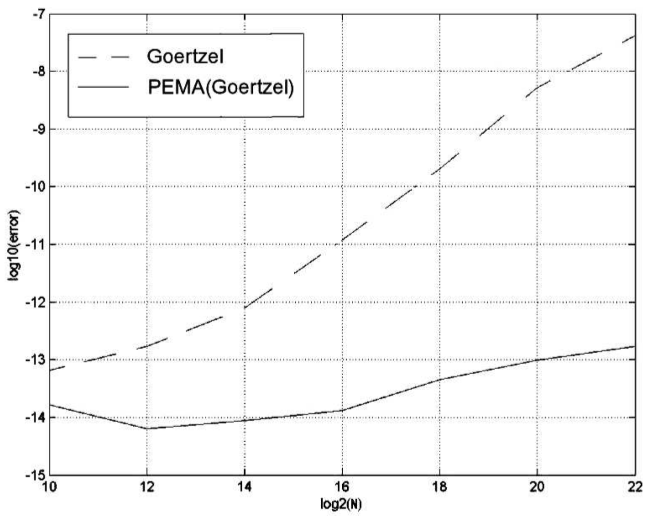

Figure 1: Relative errors of Goertzel's and PEMA algorithms
for polynomials with random coefficients.

Figure 2 describes similar results for a family of polynomials with coefficients given by the formula $a_k = f(t_k)$ where $t_k = 0.001k$, $k = 0, \ldots, N$ and $f(t) = sint + sin100t + sin1000t$. As before the lower and the upper graphs represent results given by PEMA and the standard Goertzel's algorithm, respectively.

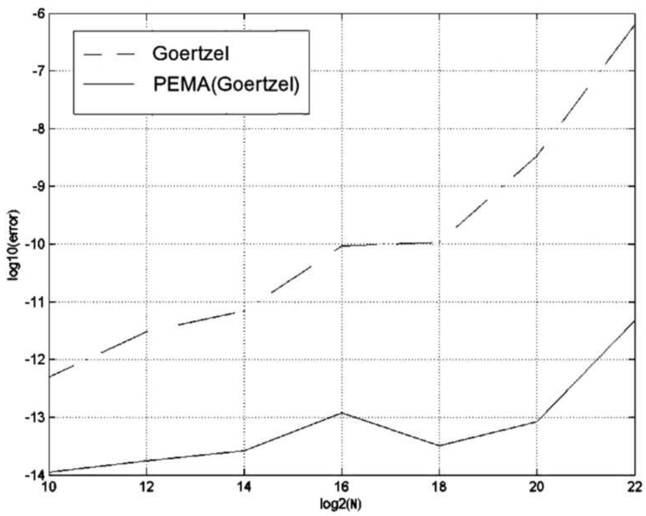

Figure 2: Relative errors of Goertzel's and PEMA algorithms
for polynomials with coefficients $a_k = f(t_k)$
where $f(t) = sint + sin100t + sin1000t$.

Figure 3 illustrates analogous results for polynomials with coefficients

$a_k = \sqrt{k}$. And again the lower graph represents results given by PEMA.

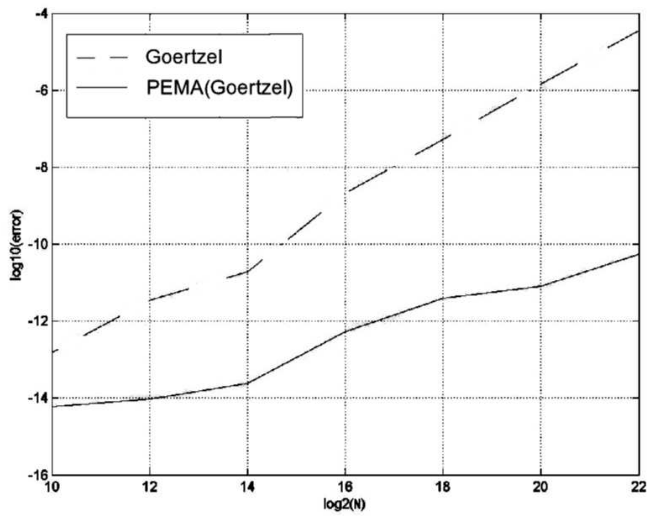

Figure 3: Relative errors of Goertzel's and PEMA algorithms for polynomials with coefficients $a_k = \sqrt{k}$.

Tables $1-3$ contain values of $error$ (16) for each method and for polynomials of coefficients given in description above each table. N is the polynomial degree. The second and the third columns contain results given by Horner's rule and the version of PEMA algorithm with Horner's rule, respectively. Data in the last two columns is results given by Goertzel's algorithm and PEMA with Goertzel's algorithm. This data was used to create figures $1-3$.

Note that although Goertzel's algorithm gives large errors for large N, PEMA using Goertzel's algorithm has much smaller errors; they are comparable with the errors obtained using Horner's algorithm, or PEMA with Horner's algorithm.

Table 1: Relative errors of Goertzel's, Horner's and both versions of PEMA algorithms for polynomials with random coefficients.

N	Horner	PEMA(Horner)	Goertzel	PEMA(Goertzel)
2^{10}	$1.6396e-014$	$1.6597e-014$	$6.4827e-014$	$1.6614e-014$
2^{12}	$6.4839e-015$	$6.2312e-015$	$1.7241e-013$	$6.3318e-015$
2^{14}	$6.4597e-015$	$8.8147e-015$	$7.8870e-013$	$8.8450e-015$
2^{16}	$1.0575e-014$	$1.2730e-014$	$1.1884e-011$	$1.3035e-014$
2^{18}	$3.0060e-014$	$4.3985e-014$	$2.0332e-010$	$4.4917e-014$
2^{20}	$7.1352e-014$	$7.6212e-014$	$5.2591e-009$	$9.7373e-014$
2^{22}	$1.1814e-013$	$1.5060e-013$	$4.1586e-008$	$1.7229e-013$

Table 2: Relative errors of Goertzel's, Horner's and both versions of PEMA algorithms for polynomials with coefficients $a_k = f(t_k)$ where $f(t) = sint + sin100t + sin1000t$.

N	Horner	PEMA(Horner)	Goertzel	PEMA(Goertzel)
2^{10}	$2.1321e-015$	$1.0999e-014$	$4.9016e-013$	$1.1313e-014$
2^{12}	$4.3372e-015$	$1.5549e-014$	$3.0108e-012$	$1.7462e-014$
2^{14}	$9.7481e-015$	$2.5365e-014$	$7.0483e-012$	$2.6262e-014$
2^{16}	$3.2760e-014$	$1.0139e-013$	$9.2595e-011$	$1.1973e-013$
2^{18}	$1.6703e-014$	$1.6408e-014$	$1.0737e-010$	$3.2052e-014$
2^{20}	$8.1798e-014$	$6.0448e-014$	$3.3356e-009$	$8.3002e-014$
2^{22}	$2.6576e-011$	$3.9179e-011$	$6.4574e-006$	$4.8041e-011$

Table 3: Relative errors of Goertzel's, Horner's and both versions of PEMA algorithms for polynomials with coefficients $a_k = \sqrt{k}$.

N	Horner	PEMA(Horner)	Goertzel	PEMA(Goertzel)
2^{10}	$5.6281e-015$	$5.6566e-015$	$1.5038e-013$	$5.9073e-015$
2^{12}	$8.0767e-015$	$8.1583e-015$	$3.4601e-012$	$9.3555e-015$
2^{14}	$1.8735e-014$	$1.8795e-014$	$1.9228e-011$	$2.3707e-014$
2^{16}	$1.7620e-013$	$4.7930e-013$	$2.1008e-009$	$5.2504e-013$
2^{18}	$1.1682e-012$	$3.5980e-012$	$5.1238e-008$	$3.8532e-012$
2^{20}	$8.4972e-012$	$6.1673e-012$	$1.4374e-006$	$8.1276e-012$
2^{22}	$5.1749e-011$	$4.1890e-011$	$3.5824e-005$	$5.3874e-011$

Appendix. Error analysis of Goertzel's algorithm

Now we turn our attention to numerical analysis of Goertzel's algorithm. Goertzel's method is a special case of Clenshaw's algorithm (Cf. [1], [2], [10]). Our results are similar in spirit to these given by Gentleman [2], who gave a floating-point error analysis of Goertzel's algorithm for computing Fourier coefficients $\sum_{n=0}^{N} a_n \cos n\xi$ and $\sum_{n=1}^{N} a_n \sin n\xi$ with respect to the data $a_0, \ldots, a_N$ and a given argument ξ (Cf. [11], pp. 84-88, [2]). He advised to avoid this technique, particularly for low frequencies ξ (e.g. for $\xi = 0$). However, we prove that under natural assumptions Goertzel's algorithm is numerically stable in a sense (2), as an algebraic polynomial evaluation algorithm. These results extend the results obtained in [2], [9] for real coefficients a_n. Here we consider more general case of complex coefficients a_n.

In the exact arithmetic we have for the quantities computed by Goertzel's algorithm (Algorithm 2)

$$b_n = \sum_{k=n}^{N} a_k \, |z|^{k-n} \; U_{k-n}(t), \quad n = 1, 2, \ldots, N, \tag{17}$$

$$u = \sum_{k=0}^{N} a_k |z|^k \; T_k(t), \quad v = y \; \sum_{k=1}^{N} a_k |z|^{k-1} \; U_{k-1}(t), \tag{18}$$

$$t = \frac{x}{|\,z\,|}, \ \ t \in [-1, 1] \tag{19}$$

and $T_k(t)$ and $U_k(t)$ are the Chebyshev polynomials of the first kind and the second kind, respectively. They satisfy the recurrence relations (Cf. [12])

$$T_k(t) = 2tT_{k-1}(t) - T_{k-2}(t), \ \ U_k(t) = 2tU_{k-1}(t) - U_{k-2}(t), \quad k = 2, \ldots$$

with $T_0(t) = U_0(t) = 1$ and $T_1(t) = t, \ U_1(t) = 2t$.

Moreover,

$$T_k(t) = tU_{k-1}(t) - U_{k-2}(t), \quad k = 2, 3, \ldots$$

We remind very well known inequalities for $t \in [-1, 1]$:

$$|T_k(t)| \leq 1, \ \ |U_k(t)| \leq k + 1, \quad k = 0, 1, \ldots \tag{20}$$

It's well known [12] that for $|t| < 1$

$$T_k(t) = \cos k\theta, \quad U_k(t) = \frac{\sin(k+1)\theta}{\sin\theta},$$

where

$$t = \cos\theta, \quad z = |z|e^{i\theta}, \quad \theta \in (0, \pi).$$

Notice that

$$\frac{z^k}{|z|^k} = T_k(t) + i\frac{y}{|y|}U_{k-1}(t) \quad \text{for} \quad k = 0, 1, \ldots, N. \tag{21}$$

Now we analyze numerical behaviour of Goertzel's algorithm in floating-point arithmetic.

Let $\tilde{w}(z) = \tilde{u} + i\tilde{v}$, $\tilde{b}_n$, $\tilde{p}, \tilde{q}$ denote the quantities computed numerically in cfl (see section 3.2). We have

$$\tilde{p} = p, \ \ \tilde{q} = q(1 + \gamma), \ \ |\gamma| \leq 2\epsilon_M + \mathcal{O}(\epsilon_M^2).$$

Therefore, $\tilde{b}_{N+1} = 0, \ \tilde{b}_N = a_N$ and for $n = N-1, \ldots, 1$ we get

$$\tilde{b}_n = (a_n + \eta_n) + p\,\tilde{b}_{n+1} + q\,\tilde{b}_{n+2}, \tag{22}$$

$$\tilde{u} = (a_0 + \eta_0) + x\,\tilde{b}_1 + q\,\tilde{b}_2 \tag{23}$$

where for $n = 0, 1, \dots, N$

$$|\eta_n| \le K\,\epsilon_M\,(|a_n| + |z||\tilde{b}_{n+1}| + |z|^2|\tilde{b}_{n+2}|), \quad K = 5 + \mathcal{O}(\epsilon_M). \tag{24}$$

The constant 5 in (24) is overestimated, but this way error analysis is simpler and the essential result is the same.

Further, we get

$$\tilde{w}(z) = (\tilde{u} + i\,\tilde{v})\,(1 + \delta_2), \quad \tilde{v} = y\,\tilde{b}_1(1 + \delta_1), \quad |\delta_1|, |\delta_2| \le \epsilon_M. \tag{25}$$

From this it follows that

$$\tilde{b}_n = \sum_{k=n}^{N} (a_k + \eta_k)|z|^{k-n}\,U_{k-n}(t), \;\; n = 1, 2, \dots, N \tag{26}$$

and

$$\tilde{u} = \sum_{k=0}^{N} (a_k + \eta_k)|z|^k\,T_k(t). \tag{27}$$

From (17), (18) and (20) it follows that

$$|u| \le g_0, \;\; |b_n| \le (N - n + 1)g_n, \quad n = 1, \dots, N, \tag{28}$$

where

$$g_n = \sum_{k=n}^{N} |a_k|\,|z|^{k-n}, \quad n = 0, 1, \dots, N. \tag{29}$$

We want to estimate the absolute error $|\tilde{w}(z) - w(z)|$. Let's write $\tilde{b}_n$ as $\tilde{b}_n = b_n + e_n$, where from (20), (26) and (27)

$$|e_n| \le (N - n + 1)\sum_{k=n}^{N} |\eta_k|\,|z|^k. \tag{30}$$

The formulae (24), (26)-(29) yield

$$|\eta_k| \le K\,\epsilon_M\,(|a_k| + (N-k)|z|g_{k+1} + (N-k-1)|z|^2 g_{k+2}) + \mathcal{O}(\epsilon_M^2).$$

Thus

$$|\eta_k| \le 2K\,\epsilon_M\,(N-k)g_k + \mathcal{O}(\epsilon_M^2), \quad \text{for} \quad k = 0, 1, \ldots, N. \tag{31}$$

Now write analogously $\tilde{u} = u + e_0$, where $|e_0| \le \sum_{k=0}^{N} |\eta_k|\,|z|^k$.
It's easy to check that $\sum_{k=0}^{N} g_k\,|z|^k \le (N+1)\,g_0$. From this and (31) we get

$$\sum_{k=0}^{N} |\eta_k|\,|z|^k \le 2K(N+1)N\epsilon_M g_0 + \mathcal{O}(\epsilon_M^2). \tag{32}$$

Now let's rewrite (25) as

$$\tilde{w}(z) = (\tilde{u} + i\,y\,\tilde{b}_1) + \xi. \tag{33}$$

It's easy to verify that

$$|\xi| \le (2N+1)\epsilon_M g_0 + \mathcal{O}(\epsilon_M^2). \tag{34}$$

Further from (21), (26) and (27) we get

$$\tilde{u} + i\,y\,\tilde{b}_1 = \sum_{k=0}^{N} (a_k + \eta_k)\,z^k.$$

This and (33) yield

$$|\tilde{w}(z) - w(z)| \le \sum_{k=n}^{N} |\eta_k|\,|z|^k + |\xi|.$$

Combining this with (32) and (34) we get the inequality

$$|\tilde{w}(z) - w(z)| \le 2K(N+1)^2\epsilon_M g_0 + \mathcal{O}(\epsilon_M^2),$$

which can be reformulated in the following

Theorem 2 *Assume that $cfl(z) = z$ and $cfl(a_n) = a_n$ for $n = 0, \ldots, N$. Let*

$$2K(N+1)^2\,\epsilon_M \le 0.1, \tag{35}$$

where K is defined in (24).
Then Goertzel's algorithm for computing $w(z) = \sum_{n=0}^{N} a_n z^n$ is componentwise backward stable, i.e.

$$\tilde{w}(z) = \sum_{n=0}^{N} a_n (1 + \Delta_n)\, z^n, \qquad |\Delta_n| \leq A_N\, \epsilon_M + \mathcal{O}(\epsilon_M^2), \tag{36}$$

where

$$A_N = 2K(N+1)^2. \ \blacksquare \tag{37}$$

Notice that $A_N \approx 10N^2$. Numerical tests in section 4 confirm that the constant N^2 is realistic.

References

[1] C. W. Clenshaw, *A note on the summation of Chebyshev series*, Math. Tables Aids Comput. 9 (1955), pp. 118–120.

[2] W. M. Gentleman, *An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients*, Comput. J. 12 (1969), pp. 160–165.

[3] G. H. Golub, J. M. Ortega, *Scientific computing. An introduction with parallel computing*, Academic Press, New York 1993.

[4] M. Jankowski, A. Smoktunowicz and H. Woźniakowski, *A note on floating-point summation of very many terms*, Journal of Information Processing and Cybernetics-EIK 19 (1983), pp. 435-440.

[5] A. Kiełbasiński, K. Ziętak, *Numerical analysis of standard problems with a unitary Householder transformation*, Mat. Stos. (III) 8 (1976), pp. 67-80 (in Polish).

[6] Charles F. Van Loan, *Computational frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, PA, USA 1992.

[7] A. C. R. Newbery, *Error analysis for Fourier series evaluation*, Math. Comp. 27 (123) (1973), pp. 639-644.

[8] J. Oliver, *An error analysis of the modified Clenshaw method for evaluating Chebyshev and Fourier series*, J. Inst. Math. Applics. 20 (1977), pp. 379-391.

[9] A. Smoktunowicz, *Stability issues for special algebraic problems*, Ph.D. Thesis, Univ. of Warsaw, 1981 (in Polish).

[10] A. Smoktunowicz, *Backward stability of Clenshaw's algorithm*, BIT vol. 42, no. 3 (2002), pp. 600-610.

[11] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1992.

[12] G. Szegö, *Orthogonal polynomials*, rev. ed. New York, 1959.

[13] J. H. Wilkinson, *Rounding errors in algebraic processes.* Notes on Applied Science No. 32, Her Majesty's Stationary Office, London, 1963.