On the accuracy and stability of algorithms most commonly used in the evaluation of Chebyshev polynomials of the first kind

Alicja Smoktunowicz * Agata Smoktunowicz †
Ewa Pawelec ‡

December 23, 2013

Abstract

This paper provides error analyses of the algorithms most commonly used for the evaluation of the Chebyshev polynomial of the first kind $T_N(x)$. Some of these algorithms are shown to be backward stable. This means that the computed value of $T_N(x)$ in floating point arithmetic by these algorithms can be interpreted as a slightly perturbed value of polynomial $T_N$, for slightly perturbed value of $x$.

Keywords Chebyshev polynomials, roots of polynomials, error analysis

Mathematics Subject Classification (2000) 65G50, 65D20, 65L70

*Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, Warsaw, 00-662 Poland, e-mail: smok@mini.pw.edu.pl
†School of Mathematics, University of Edinburgh, Edinburgh, Scotland EH9 3JZ, UK, e-mail: A.Smoktunowicz@ed.ac.uk. The research of Agata Smoktunowicz was funded by ERC grant 320974.
‡Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, Warsaw, 00-662 Poland, e-mail: E.Pawelec@mini.pw.edu.pl
1 Introduction

Chebyshev polynomials of the first kind \((T_n(x))\) are widely used in many applications. They satisfy the three-term recurrence

\[
T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad n = 2, 3, \ldots ,
\]

(1)

where \(T_0(x) = 1, T_1(x) = x\).

There are several algorithms for evaluating \(T_N(x)\) (see [2], [3], [7], [9]). However, for numerical purposes some of them are poor (see [1], [6], [7]). For example, using the symbolic calculations in MATHEMATICA, MAPLE, DERIVE and others packages, it is possible to find the expanded form of \(T_N(x)\), that is, the exact coefficients \(a_n\) of \(T_N(x)\) such that \(T_N(x) = a_0 + a_1 x + \cdots + a_N x^N\). However computing the value \(T_N(x)\) at a given floating point \(x\) from this form can be disastrous. At first this may seem surprising, since the coefficients \(a_n\) are integers. Note that there are large \(a_n\) for large \(N\), for example the leading coefficient \(a_N = 2^{N-1}\).

Symbolic and numeric computations often demand different approaches (see [7]). In practice, a desirable property for an algorithm is numerical stability (see [11]). Our problem of computing the value \(T_N(x)\) at a given point \(x\) is a special case of the general problem of evaluating the polynomial \(p_N(x) = c_0 T_0(x) + c_1 T_1(x) + \cdots + c_N T_N(x)\). Clenshaw’s and Forsythe’s algorithms are recommended here. An error analysis of Clenshaw’s algorithm in the general case was first provided by D. Elliott in [5]. See also [1], [6], [9], [2], [3], where the authors gave the forward error bounds for the evaluation of \(p_N(x)\) in floating point arithmetic. However, it is of interest to know whether an algorithm is backward stable with respect to the data \(x\). Roughly speaking, the computed value \(\tilde{T}_N(x)\) by a backward stable algorithm can be interpreted as a slightly perturbed value of the polynomial \(T_N\) for a slightly perturbed value of \(x\). A more precise definition is now given.

**Definition 1** An algorithm \(W\) of computing \(T_N(x)\) is backward stable with respect to the data \(x\) if the value \(\tilde{T}_N(x)\) computed by \(W\) in floating point
arithmetic satisfies

\[ \tilde{T}_N(x) = (1 + \delta_N)T_N((1 + \Delta_N)x) + O(\epsilon_M^2), \quad |\delta_N|, |\Delta_N| \leq \epsilon_M L, \]  

(2)

where \( L = L(N) \) is a modest constant and \( \epsilon_M \) is machine precision.

Throughout this paper we will ignore the terms of order \( O(\epsilon_M^2) \). It is easy to check that (2) is equivalent to

\[ |\tilde{T}_N(x) - T_N(x)| \leq \epsilon_M L C_N(x) + O(\epsilon_M^2), \]  

(3)

where

\[ C_N(x) = |T_N(x)| + |xT'_N(x)|. \]  

(4)

Note that

\[ C_N(x) = |T_N(x)| + N|xU_{N-1}(x)|, \]  

(5)

where \( U_{N-1}(x) \) denotes the Chebyshev polynomial of the second kind. These polynomials satisfy the recurrence relations

\[ U_n(x) = 2xU_{n-1}(x) - U_{n-2}(x), \quad n = 2, 3, \ldots, \]  

(6)

where \( U_0(x) = 1, \ U_1(x) = 2x. \)

We will consider the following algorithms for computing \( T_N(x) \) at a given point \( x \in [-1, 1] \).

- **Algorithm I (Three – term recursion)**

  \[ \begin{align*}
  T_0 &= 1; & T_1 &= x; \\
  T_n &= 2xT_{n-1} - T_{n-2} & \text{for } n = 2, 3, \ldots, N. \\
  T_N(x) &= T_N.
  \end{align*} \]

- **Algorithm II (Fast)**

  Let \( N = 2^p \).

  This algorithm uses the identity \( T_{2n}(x) = T_2(T_n(x)) \) and computes \( R_n = T_{2^n}(x) \) as follows:

  \[ \begin{align*}
  R_0 &= x; \\
  R_n &= 2R_{n-1}^2 - 1 & \text{for } n = 1, \ldots, p. \\
  T_N(x) &= R_p.
  \end{align*} \]
- **Algorithm III (Trigonometric)**
  \[ T_N(x) = \cos(N \ast \arccos(x)). \]

- **Algorithm IV (Horner)**
  Use Horner’s scheme for the expanded form of \( T_N(x) \):
  \[ T_N(x) = 2^{N-1}x^N + a_{N-1}x^{N-1} + \ldots + a_0. \]
  Note that the coefficients \( a_n \) are integers.

The rest of this paper is organized as follows. In Section 2 we recall some basic properties of the Chebyshev polynomials. In Section 3 we will use these properties in a derivation of the lower and upper bounds for \( C_n(x) \). In Section 4 we present the error analyses for Algorithms I and II above, proving that these algorithms are backward stable in the sense of \([9]\). In Section 5 we compare the accuracy of the algorithms using numerical experiments performed in MATLAB; our tests show that Algorithm III can be less accurate for \( x \) near \( \pm 1 \) and that Algorithm IV is not always backward stable.

## 2 Preliminaries

We will need some properties of the Chebyshev polynomials (see \([8]\) and \([10]\)). For \(-1 \leq x \leq 1\) we have \( T_n(x) = \cos(n \Theta) \), where \( \Theta = \arccos x \) and \( U_{n-1}(x) = \sin(n \Theta)/\sin \Theta \) for \( 0 < x < 1 \).

The following identities hold

\[
U_{n-1}(x) = \frac{T'_n(x)}{n},
\]

\[
T_n(-x) = (-1)^n T_n(x), \quad U_n(-x) = (-1)^n U_n(x).
\]

The Chebyshev polynomials of the first kind satisfy the following differential equations

\[
(1 - x^2)T''_n(x) - xT'_n(x) + n^2T_n(x) = 0
\]  

\[ (7) \]
and
\[ T_n^2(x) + \frac{1-x^2}{n^2} T_n^2(x) = 1. \] (8)

The last equality is a consequence of the trigonometric identity \( \cos^2 n\theta + \sin^2 n\theta = 1. \)

For \(-1 \leq x \leq 1\) and \(n = 0, 1, \ldots\) we have the upper bounds
\[ |T_n(x)| \leq |T_n(1)| = 1, \quad |U_n(x)| \leq |U_n(1)| = n + 1 \] (9)

and for \(-1 < x < 1\)
\[ |U_n(x)| \leq \frac{1}{\sqrt{1-x^2}} \] (10)

The roots \((t_i)\) of \(T_n(x)\) are distinct and belong to \((-1, 1)\):
\[ t_i = \cos \left( \frac{2i-1}{2n} \pi \right), \quad i = 1, 2, \ldots, n. \] (11)

The roots \((u_i)\) of \(T'_n(x)\) (i.e. the roots of \(U_{n-1}(x)\)) are:
\[ u_i = \cos \left( \frac{i\pi}{n} \right), \quad i = 1, 2, \ldots, n-1. \] (12)

Then \(-1 < t_n < u_{n-1} < \ldots < u_1 < t_1 < 1\) and
\[ T_n(u_i) = (-1)^i \quad i = 1, 2, \ldots, n-1. \] (13)

For \(-1 \leq x \leq 1\) and \(m = 0, 1, \ldots\) we get
\[ |T_{2m+1}(x)| \leq (2m+1)|x|, \quad |U_{2m+1}(x)| \leq 2(m+1)|x|. \] (14)

In evaluating the Chebyshev polynomials one can use the composition identity
\[ T_{mn}(x) = T_m(T_n(x)), \quad m, n = 0, 1, \ldots. \] (15)

3 Lower and upper bounds for \(C_n(x)\)

Since \(C_n(-x) = C_n(x)\) for all \(x\), we restrict our considerations to the interval \([0, 1]\). From (8) it follows that \(C_n(x) \leq C_n(1) = n^2 + 1\) for \(0 \leq x \leq 1\). By (12) we have \(C_n(u_i) = 1\) for \(i = 1, \ldots, n-1\). If \(n\) is odd then \(C_n(0) = 0.\)
Theorem 1  Let \( n \) be a natural number. Assume that \( s_n \leq x \leq 1 \), where
\[
s_n = \frac{1}{\sqrt{n^2 + 1}}. \tag{16}
\]
Then we have
\[
C_n(x) = |T_n(x)| + |xT'_n(x)| \geq 1. \tag{17}
\]

Proof. Notice that the inequality \( x^2 \geq s_n^2 \) is equivalent to \( x^2 \geq \frac{1-x^2}{n^2} \). From this and (8) we get
\[
C_n^2(x) \geq T_n^2(x) + x^2T'_n^2(x) \geq T_n^2(x) + \frac{1-x^2}{n^2} T_n^2(x) = 1. \tag{18}
\]
The proof is now complete. \( \square \)

Theorem 2  Let \( n \) be a natural number. Assume that \( 0 \leq x \leq s_n \), where \( s_n \) is defined by (16). Then

(i) \( C_n(x) \geq n|x| \) for all \( n \),

(ii) \( C_n(x) \geq 1 \) for even \( n \).

Proof. We consider case (i). Clearly, \( 1 \geq n^2 x^2 \), by (16) and since \( 0 \leq x \leq s_n \). Therefore,
\[
C_n^2(x) \geq 1T_n^2(x) + x^2T'_n^2(x) \geq n^2 x^2T_n^2(x) + x^2T'_n^2(x) \geq x^2n^2(T_n^2(x) + \frac{1}{n^2} T_n^2(x)).
\]
Since \( 1 \geq 1 - x^2 \) we get
\[
C_n^2(x) \geq x^2n^2(T_n^2(x) + \frac{1-x^2}{n^2} T_n^2(x)) = x^2n^2,
\]
due to (8). Therefore, \( C_n(x) \geq n|x| \). This completes the proof of case (i).

Now we consider case (ii). Let \( n = 2m \). We first prove that \( T_{2m} \) has no roots in \((0, s_{2m})\). By (14), we need to show that
\[
t_m = \cos \left( \frac{(2m - 1)\pi}{4m} \right) > s_{2m}. \tag{18}
\]
Notice that
\[ t_m = \cos\left(\frac{\pi}{2} - \frac{\pi}{4m}\right) = \sin\frac{\pi}{4m}. \]

Since \(0 < \tan \Theta > \Theta\) for all \(0 < \Theta < \frac{\pi}{2}\), we have \(\tan^2 \Theta > \frac{\Theta^2}{1 + \Theta^2}\). From this it follows that \(\sin^2 \Theta > \frac{\Theta^2}{1 + \Theta^2}\). Substituting \(\Theta = \pi/4m\) in the above inequality leads to
\[ t_m^2 > \frac{\pi^2}{16m^2 + \pi^2} > \frac{1}{4m^2 + 1} = s_{2m}^2, \]
so \(t_m > s_{2m}\). This finishes the proof of (18).

We see that \(T_{2m}\) has no roots in \((0, s_{2m})\). Moreover, \(T_{2m}(0) = (-1)^m\) and \(T'_{2m}(0) = 0\). We conclude from (11)–(12) that 0 is the only root of \(T_{2m}'\) in the interval \((-s_{2m}, s_{2m})\). Notice that \(T_{2m}\) and \(T''_{2m}\) are even, i.e. \(T_{2m}(-x) = T_{2m}(x)\) and \(T''_{2m}(-x) = T''_{2m}(x)\) for all \(x\). \(T_{2m}'\) is odd, that is, \(T'_{2m}(-x) = -T'_{2m}(x)\). Thus we see that the polynomials \(T_{2m}\) and \(T_{2m}'\) do not change the signs in \((0, s_{2m})\).

More precisely, if \(m\) is even, then for all \(0 < x < s_{2m}\) we have \(T_{2m}(x) > 0\) and \(T'_{2m}(x) < 0\), hence \(C_{2m}(x) = T_{2m}(x) - xT'_{2m}(x)\). Similarly, if \(m\) is odd then \(T_{2m}(x) < 0\) and \(T'_{2m}(x) > 0\), so \(C_{2m}(x) = -T_{2m}(x) + xT'_{2m}(x)\). We see that \(C'_{2m}(x) = -T''_{2m}(x)\) if \(m\) is even and \(C'_{2m}(x) = T''_{2m}(x)\) otherwise.

By (7) for \(n = 2m\), we obtain the formula
\[ (1 - x^2)T''_{2m}(x) = xT'_{2m}(x) - 2m^2T_{2m}(x). \]
We see that for all \(0 < x < s_{2m}\) we have \(T''_{2m}(x) < 0\) if \(m\) is even and \(T''_{2m}(x) > 0\) if \(m\) is odd. We conclude that \(C''_{2m}(x) > 0\) for any \(m\), so \(C_{2m}(x)\) is increasing in the interval \((0, s_{2m})\). This gives the lower bound \(C_{2m}(x) \geq C_{2m}(0) = 1\). The proof of our theorem is now complete.

4 Error analysis

As a direct consequence of Theorems 1–2 we obtained the following result.
Corollary 4.1 Let $N \geq 2$ and $s_N = \frac{1}{\sqrt{N+1}}$. Assume that an algorithm $W$ evaluates $T_N(x)$ in floating point arithmetic with the small forward error
\[ |\tilde{T}_N(x) - T_N(x)| \leq \epsilon_M L_1 + O(\epsilon_M^2), \]
where $L_1 = L_1(N)$ is a modest constant and $\epsilon_M$ is machine precision. Then
(i) if $N$ is even then $W$ is backward stable in $[-1,1]$, i.e. \( (19) \) holds with the constant $L = L_1$,
(ii) if $N$ is odd then $W$ is backward stable for $s_N \leq |x| \leq 1$ with the constant $L = L_1$,
(iii) if $N$ is odd and there is a small constant $L_2 = L_2(N)$ such that for $|x| \leq s_N$ we have
\[ |\tilde{T}_N(x) - T_N(x)| \leq \epsilon_M L_2 |x| + O(\epsilon_M^2), \]
then $W$ is backward stable for $|x| \leq s_N$ with the constant $L = L_2/N$.

4.1 Error analysis of Algorithm I

We analyze the rounding errors in Algorithm I.

Theorem 3 Let $N \geq 2$ and $s_N = \frac{1}{\sqrt{N+1}}$. Let $\tilde{T}_n$ denote the quantities computed by Algorithm I in floating point arithmetic $fl$ with machine precision $\epsilon_M$. Let $\tilde{T}_N(x) = \tilde{T}_N$. Assume that $x$ is exactly representable in $fl$ ($fl(x) = x$) and $x \in [-1,1]$.

Then we have the bound
\[ |\tilde{T}_N(x) - T_N(x)| \leq \epsilon_M \frac{3N(N-1)}{2} + O(\epsilon_M^2). \]
If $|x| \leq s_N$ then
\[ |\tilde{T}_N(x) - T_N(x)| \leq \epsilon_M \frac{9(N-1)}{2} + O(\epsilon_M^2). \]
Moreover, if $|x| \leq s_N$ and $N$ is odd then
\[ |\tilde{T}_N(x) - T_N(x)| \leq \epsilon_M \frac{5(N-1)(N+7)}{8} |x| + O(\epsilon_M^2). \]
Proof. Note that $\tilde{T}_0 = 1$, $\tilde{T}_1 = x$ and for $n = 2, \ldots$ we have

$$\tilde{T}_n = (2x \tilde{T}_{n-1}(1 + \alpha_n) - \tilde{T}_{n-2})(1 + \beta_n), \quad |\alpha_n|, |\beta_n| \leq \epsilon_M.$$ 

We rewrite it as follows

$$\tilde{T}_n = 2x \tilde{T}_{n-1} - \tilde{T}_{n-2} + \xi_n, \quad \xi_n = 2x \tilde{T}_{n-1} \alpha_n + \frac{\beta_n}{1 + \beta_n} \tilde{T}_n. \quad (24)$$

Let $e_n = \tilde{T}_n - T_n(x)$. We observe that $e_0 = e_1 = 0$ and $e_n = 2xe_{n-1} - e_{n-2} + \xi_n$ for $n = 2, 3, \ldots, N$. From this it follows that

$$e_N = \tilde{T}_N - T_N(x) = \sum_{n=2}^{N} U_{N-n}(x) \xi_n.$$ 

Therefore,

$$|e_N| \leq \sum_{n=2}^{N} |U_{N-n}(x)||\xi_n|.$$ 

This together with (24) leads to

$$|\xi_n| \leq \epsilon_M (2|x||T_{n-1}(x)| + |T_n(x)|) + \mathcal{O}(\epsilon_M^2), \quad (25)$$

hence

$$|e_N| \leq \epsilon_M \sum_{n=2}^{N} (2|x||T_{n-1}(x)| + |T_n(x)|)|U_{N-n}(x)| + \mathcal{O}(\epsilon_M^2). \quad (26)$$

Since $|T_n(x)| \leq 1$ for $|x| \leq 1$ we obtain

$$|e_N| \leq \epsilon_M 3 \sum_{n=2}^{N} |U_{N-n}(x)| + \mathcal{O}(\epsilon_M^2). \quad (27)$$

This together with (9) leads to

$$|e_N| \leq \epsilon_M 3 \sum_{n=2}^{N} (N - n + 1) + \mathcal{O}(\epsilon_M^2) \leq \epsilon_M \frac{3N(N - 1)}{2} + \mathcal{O}(\epsilon_M^2).$$

The proof of (21) is complete.

Now consider the case $|x| \leq s_N$. By (10) we get $|U_k(x)| \leq \frac{1}{\sqrt{1-s_N}}$ for $k = 0, 1, \ldots$. 

9
Therefore,
\[ |U_k(x)| \leq \frac{3}{2} \text{ for } |x| \leq s_N, \quad k = 0, 1, \ldots \]  
(28)

From this and (27) the bound (22) follows immediately.

Now assume that \( N \) is odd and \( |x| \leq s_N \). We rewrite (26) as follows
\[ |e_N| \leq \epsilon_M (A_N(x) + B_N(x)) + \mathcal{O}(\epsilon_M^2), \]  
(29)

where
\[ A_N(x) = 2|x| \sum_{n=2}^{N} |T_{n-1}(x)||U_{N-n}(x)|, \]  
(30)
\[ B_N(x) = \sum_{n=2}^{N} |T_n(x)||U_{N-n}(x)|. \]  
(31)

This together with (28) and the inequality \(|T_{n-1}(x)| \leq 1\) gives
\[ A_N(x) \leq 3|x|(N - 1). \]  
(32)

To estimate \( B_N(x) \) for \( N = 2m + 1 \) we split it as follows
\[ B_N(x) = \sum_{k=1}^{m} |T_{2k}(x)||U_{N-2k}(x)| + \sum_{k=1}^{m} |T_{2k+1}(x)||U_{N-(2k+1)}(x)|. \]

Note that (14) implies the following upper bounds (for the polynomials of the odd degrees)
\[ |U_{N-2k}(x)| \leq (N - 2k + 1) |x|, \quad |T_{2k+1}(x)| \leq (2k + 1) |x|. \]

By (28), we have \(|U_{N-(2k+1)}(x)| \leq \frac{3}{2} |x|\) for \(|x| \leq s_N\). We conclude that
\[ B_N(x) \leq \left( \sum_{k=1}^{m} 1(N - 2k + 1) |x| + \frac{3}{2} \sum_{k=1}^{m} (2k + 1) |x| \right). \]

The last inequality together with (29) and (32) leads to
\[ |e_N| \leq \epsilon_M (3(N - 1) + m(N - m) + \frac{3}{2} m(m + 2)) |x| + \mathcal{O}(\epsilon_M^2). \]

Since \( m = (N - 1)/2 \) we get immediately (23). \( \Box \)

By Corollary 4.1 we conclude that Algorithm I is backward stable in \([-1, 1]\) with the constant \( L \) of order \( N^2 \). Algorithm I is backward stable with the constant \( L \) of order \( N \) for \(|x| \leq s_N\).
4.2 Error analysis of Algorithm II

Theorem 4 Let \( N = 2^p \) and \( \tilde{R}_n \) denote the quantities computed by Algorithm II in floating point arithmetic \( \text{fl} \) with machine precision \( \epsilon_M \). Let \( \tilde{T}_N(x) = \tilde{R}_p \). Assume that \( \text{fl}(x) = x \) and \( x \in [-1,1] \).

Then
\[
|\tilde{T}_N(x) - T_N(x)| \leq \epsilon_M N^2 + \mathcal{O}(\epsilon_M^2) \tag{33}
\]
and \( (3) \) holds with the constant \( L = N^2 \).

Proof. We see that \( \tilde{R}_0 = x \) and for \( n = 1,2,\ldots,p \) we have
\[
\tilde{R}_n = (2 \tilde{R}_{n-1}^2(1 + \alpha_n) - 1)(1 + \beta_n), \quad |\alpha_n|, |\beta_n| \leq \epsilon_M.
\]
From this it follows that
\[
\tilde{R}_n = 2 \tilde{R}_{n-1}^2 - 1 + \xi_n, \quad \xi_n = 2 \tilde{R}_{n-1}^2 \alpha_n + \frac{\beta_n}{1 + \beta_n} \tilde{R}_n. \tag{34}
\]
We can prove by induction on \( n \) that
\[
\tilde{R}_n - R_n = \sum_{k=1}^{n-1} 4^{n-k} T_{2^k}(x) T_{2^{k+1}}(x) \cdots T_{2^{n-1}}(x) \xi_k + \xi_n + \mathcal{O}(\epsilon_M^2).
\]
Since \( |T_k(x)| \leq 1 \) for \( x \in [-1,1] \) we obtain
\[
|\tilde{R}_n - R_n| \leq \sum_{k=1}^{n} 4^{n-k} |\xi_k| + \mathcal{O}(\epsilon_M^2).
\]
This together with \( (34) \) gives \( |\xi_k| \leq 3\epsilon_M + \mathcal{O}(\epsilon_M^2) \), so
\[
|\tilde{R}_n - R_n| \leq \epsilon_M 3 \sum_{k=1}^{n} 4^{n-k} + \mathcal{O}(\epsilon_M^2).
\]
Finally, for \( n = p \) we get the following upper bound on \( \tilde{T}_N(x) = \tilde{R}_p \)
\[
|\tilde{T}_N(x) - T_N(x)| \leq \epsilon_M N^2 + \mathcal{O}(\epsilon_M^2). \tag{35}
\]

From Corollary 4.1 we conclude that \( (3) \) holds with the constant \( L = N^2 \), so Algorithm II is backward stable.
5 Numerical tests

To illustrate our results we present numerical tests in MATLAB with machine precision $\epsilon_M = 2^{-52} \approx 2.2 \cdot 10^{-16}$. We compare the results computed by Algorithms I–IV with the exact values of the Chebyshev polynomial $T_N(x)$. They were obtained by implementing Algorithm I in high precision using the VPA (Variable Precision Arithmetic) function from MATLAB’s Symbolic Math Toolbox and then rounded to 16th decimal digits. We compute the relative error

$$e_N = \frac{\max_{x \in S} |T_N(x) - \tilde{T}_N(x)|}{\epsilon_M}. \quad (36)$$

Here $S$ consists of $p$th equally spaced checkpoints $t_1, t_2, \ldots, t_p$ from the interval $[a, b]$, where $-1 \leq a < b \leq 1$, i.e. $t_i = a + (i - 1)h$, $i = 1, 2, \ldots, p$ and $h = (b - a)/(p - 1)$.

Table 1: The error (36) for Algorithms I–IV in $[-1, 1]$ and $h = 1/100$.

| $N$ | Algorithm I | Algorithm II | Algorithm III | Algorithm IV |
|-----|-------------|--------------|---------------|--------------|
| 8   | 5.25        | 6.68         | 13.12         | 95.68        |
| 16  | 11.00       | 12.00        | 22.37         | 3.48e+04     |
| 32  | 21.78       | 43.00        | 55.12         | 3.13e+10     |
| 64  | 35.00       | 98.75        | 88.50         | 4.83e+22     |
| 128 | 66.00       | 257.00       | 193.50        | 2.88e+47     |
| 256 | 165.00      | 888.75       | 410.25        | 1.09e+96     |
| 512 | 280.75      | 1770.0       | 841.87        | 1.61e+194    |
| 1024| 679.62      | 3570.0       | 1783.20       | NaN          |

We see that Algorithm IV is poor as a method of evaluating the Chebyshev polynomial $T_N(x)$, even for $N \geq 16$. The best results are produced by Algorithm I. These tests indicate that Algorithm II is less accurate than Algorithm I.

Table 2: The error (36) for Algorithms I and III in $[-0.8, -0.6]$ and $h = 1/1000$.
Table 3: The error for Algorithms I and III in \([-1, -0.8]\) and \(h = 1/1000\).

| \(N\) | Algorithm I | Algorithm III |
|-----|-------------|---------------|
| 100 | 35.500      | 176.125       |
| 300 | 104.125     | 607.750       |
| 500 | 164.50      | 1008.0        |
| 800 | 262.25      | 1355.0        |
| 900 | 289.50      | 2159.0        |
| 1000| 340.34      | 2137.0        |

These tests show that Algorithm III can be much less accurate than Algorithm I for \(x\) near \(-1\). Numerical properties of Algorithm III strongly depend upon the accuracy of computing the trigonometric functions \(\cos\) and \(\arccos\). For a deeper discussion of the accuracy of the evaluation of trigonometric series we refer the reader to [6].

References

[1] N. S. Bakhvalov, The stable calculation of polynomial values, J. Comp. Math. and Math. Phys. 11 (1971) 1568–1574.

[2] R. Barrio, Rounding error bounds for the Clenshaw and Forsythe algorithms for the evaluation of orthogonal series, J.Comput.Appl.Math. 138 (2002) 185–204.
[3] R. Barrio, A unified rounding error bound for polynomial evaluation, Adv. Comput. Math. 19(4) (2003) 385–399.

[4] P. Deuflhard, On algorithm for the summation of certain special functions, Computing 17 (1976) 37–48.

[5] D. Elliott, Error analysis of an algorithm for summing certain finite series, J. Austral. Math. Soc. 8 (1968) 213–221.

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

[7] W. Koepf, Efficient computation of Chebyshev polynomials, Computer Algebra Systems: A Practical Guide (Ed. M. J. Wester), New York: Wiley, 79–99 (1999).

[8] S. Paszkowski, Numerical applications of Chebyshev polynomials, Warsaw 1975 (in Polish).

[9] A. Smoktunowicz, Backward stability of Clenshaw’s algorithm, BIT 42 (3) (2002) 600–610.

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

[11] J. H. Wilkinson, The algebraic eigenvalue problems, Oxford University Press, 1965.