Accurate Evaluation of Finite Laguerre Series

Keshan He[1,a,*], Peibing Du[1,a] , Hao Jiang[1,a], Xueci Zhao[1,a], Housen Li[3,b] and Lizhi Cheng[2,a]

[1] School of Science, National University of Defense Technology, Changsha, China
[2] College of Computer, National University of Defense Technology, Changsha, China
[3] Max Planck Institute for Biophysical Chemistry, Am Fassberg 11, 37077 Geottingen, Germany
[a] {hekeshan11, dupeibing10, zhaoxueci11, Hao Jiang, Lizhi Cheng}@nudt.edu.cn,
[b] housen.li@mplibpc.mpg.del,
*corresponding author

Keywords: Compensated Algorithm, Laguerre Series, Clenshaw, Floating-Point Arithmetic, Round-off Error

Abstract. A compensated algorithm is presented to evaluate finite Laguerre series. We record round-off error in Clenshaw algorithm by using error-free transformations (EFTs). In this work, we establish a valid error estimate that is as accurate as the Clenshaw scheme using twice the working precision. Numerical experiments illustrate that our compensated algorithm is more accurate than the Clenshaw algorithm and faster than the DDClenshaw algorithm (Clenshaw in double-double arithmetic).

1. Introduction

The Generalized Laguerre polynomial is widely used in numerical analysis [1, 2, 3] and Quantum [4]. It is defined with three-term recurrence relation as follow.

$$\begin{cases} L_0^{(\alpha)}(x) &= 1, \\ L_1^{(\alpha)}(x) &= 1+\alpha-x, \\ L_{k+1}^{(\alpha)}(x) &= \dfrac{-x+2k+1+\alpha}{k+1} L_k^{(\alpha)}(x) - \dfrac{k+\alpha}{k+1} L_{k-1}^{(\alpha)}(x). \end{cases}$$

The Laguerre polynomials are orthogonal polynomials with respect to an inner product $< f, g >= \int_0^{+\infty} f(x)g(x)e^{-x}dx$. The polynomial represented in Laguerre basis is $p(x) = \sum_{j=0}^{n} a_j L_j^{(\alpha)}(x)$ where coefficients a_j are floating-point numbers.

Clenshaw Algorithm [5] is a recursive method to calculate a linear combination of Chebyshev polynomials. It can be generalized to any class of function which can be defined by a three-term recurrence relation. The relative error bound proved in literatures [6, 7] is

$$\frac{|\, p(x) - \hat{p}(x) \,|}{|\, p(x) \,|} \le cond(p, x) \times o(u), \tag{1}$$

where $\hat{p}(x)$ is the result of Clenshaw algorithm in floating-point arithmetic. When the problem is ill-conditioned, numerous compensated algorithms [8, 9, 10] based on error-free transformations [11] are proposed to evaluate different polynomials.

We propose a compensated algorithm to evaluate Laguerre series based on error-free transformation. The algorithm can obtain a more accurate result by compensating the round-off errors to the original result. We demonstrate that the forward relative error bound satisfies

$$\frac{|p(x)-\hat{p}(x)|}{p(x)} \le o(u)+cond(p,x)\times o(u^2). \tag{2}$$

This paper is organized as follows. We introduce basic preliminaries and notations in Section 2, including error-free transformations and Clenshaw algorithm. In Section 3, we present the compensated Clenshaw algorithm and analyze the forward error of Clenshaw and Compensated Clenshaw algorithm. In Section 4, we show the experimental results to indicate our algorithm is efficient and accurate.

2. Preliminaries and Notations

2.1. Basic Definitions and Notation

Throughout the paper we assume a floating-point arithmetic adhering to IEEE 754 floating-point standard [12] and no overflow nor underflow occurs. We also assume that $fl(\cdot)$ is the result of a floating-point computation. Here, the computations are produced in a floating-point arithmetic which obeys the models.

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\rho) = \frac{(x \text{ op } y)}{1+\alpha}, \quad |\rho|,|\alpha| \le u, \tag{3}$$

where op $\in\{+,-,\times,/\}$ and u is the working precision. Besides, we note $\gamma_n := nu/(1-nu) = nu+\mathcal{O}(u^2)$ and we use $\hat{a}$ to represent the computed element of a in floating-point arithmetic. Finally, an elementary classical result of error analysis states that if $|\delta_i| \le u$ then $\prod_{i=1}^{n}(1+\delta_i) = 1+\theta_n$ with $\theta_n \le \gamma_n$. $\prod_{i=1}^{n}(1+\delta_i) = 1+\theta_n$

2.2. Error-free Transformations

In this subsection we review well-known results called error-free transformations (EFTs) . Given $a, b \in \mathbb{F}$, $x = fl(a \text{ op } b) \in \mathbb{F}$, then exist y such that $a \text{ op } b = x + y$.

We show algorithms 1-3 about addition and product of two floating-point numbers proposed respectively by Knuth [13] and Dekker [14], respectively.

Algorithm 1 Sum of two floating-point numbers [14]

function $[x, y] = \texttt{Twosum}(a, b)$
$x = fl(a + b)$
$z = fl(x - a)$
$y = fl((a - (x - z)) + (b - z))$

Algorithm 2 Error-free spiting of a floation number into two parts [13]

function $[x, y] = \texttt{split}(a)$
$c = fl(factor \cdot a) \quad (factor = 2^s + 1)$
$x = fl(c - (c - a))$
$y = fl(a - x)$

Algorithm 3 Product of two floating-point numbers [13]

function $[x, y] = \texttt{TwoProd}(a, b)$
$x = fl(a \cdot b)$
$[a_1, a_2] = \text{Split}(a)$
$[b_1, b_2] = \text{Split}(b)$
$y = fl(a_2 \cdot b_2) - (((x - a_1 \cdot b_1) - a_2 \cdot b_1) - a_1 \cdot b_2)$

Algorithm 4 Product of three floating-point numbers [15]

function $[x, y] = \texttt{ThreeProd}(a, b, c)$
$[t, h] = \text{TwoProd}(a, b)$
$[x, e] = \text{TwoProd}(t, c)$
$y = fl(c \cdot h + e)$

A compensated algorithm [15] for product of three floating-point numbers in Algorithm 4 is also presented. There are three properties of ThreeProd which will be used in error analysis:

1. $|e| \le u|abc| + u|hc| \le u|abc| + u^2|abc|$;
2. $|ch + e| \le |ch| + |e| \le 2u|abc| + u^2|abc|$;
3. If $x + y = abc - \tau$ in ThreeProd algorithm, then

$$|\tau| \le \gamma_2\gamma_6|abc|, \quad |y| \le 2u|abc| + 14u^2|abc|. \tag{4}$$

2.3. Clenshaw Algorithm and Condition Number

The Clenshaw algorithm [5] of generalized Laguerre polynomial shows in Algorithm 5.

Algorithm 5 The Clenshaw algorithm of Lagurre polynomial [5]

function $p(x) = \texttt{Clenshaw}(p, x)$
$b_{j+2} = b_{j+1} = 0$
for $j = n : -1 : 0$ **do**
$b_j = -\frac{1}{j+1}xb_{j+1} + \frac{2j+\alpha+1}{j+1}b_{j+1} - \frac{j+1+\alpha}{j+2}b_{j+2} + a_j$
end for

Reviewing [16], their works put forward a general condition number for polynomial basis defined by linear recurrence. Before giving the condition number of Laguerre polynomial, we show an absolute polynomial.

Definition 1. *Let $L_k(x)$ be a Laguerre polynomial. The absolute polynomial $\tilde{L}_k^{(\alpha)}(x)$ can be defined as follow.*

$$\tilde{L}_k^{(\alpha)}(x) = \frac{x + 2k + \alpha + 1}{k+1}\tilde{L}_{k-1}^{(\alpha)}(x) + \frac{k+\alpha}{k+1}\tilde{L}_{k-2}^{(\alpha)}(x), \tag{5}$$

where $\tilde{L}_0^{(\alpha)} = 1, \tilde{L}_1^{(\alpha)} = 1 + \alpha + x$ and $|L_k^{(\alpha)}(x)| \le \tilde{L}_k^{(\alpha)}(x)$, for all $x \ge 0$.

Motivated by [10, 11, 15], we show the condition number of Laguerre polynomial series.

Definition 2. *Let $p(x) = \sum_{i=0}^{n} a_k L_k^{(\alpha)}(x)$, the relative condition number is*

$$\text{cond}(p, x) = \frac{\tilde{p}(x)}{|p(x)|} = \frac{\sum_{i=0}^{n} |a_k| \tilde{L}_k^{(\alpha)}(|x|)}{|p(x)|}, \tag{6}$$

where $L_k^{(\alpha)}(x)$ is Laguerre polynomial, $\tilde{L}_k^{(\alpha)}(x)$ is absolute polynomial.

3. Compensated Algorithm and Error Analysis

3.1. Clenshaw Algorithm and Condition Number

In this section, we analyze the round-off error of Clenshaw algorithm and present a compensated Clenshaw algorithm to compute finite Laguerre series accurately. Furthermore, we exhibit its error bound.

Previously, we divide all real coefficients into three parts as following form in order to analyze errors.

$$A = A^{(h)} + A^{(l)} + A^{(m)}, \tag{7}$$

Where $A^{(h)}, A^{(l)} \in \mathbb{F}, A^{(m)} \in \mathbb{R}, A^{(m)} \le uA^{(l)}, A^{(l)} \le uA^{(h)}$.

Actually, in floating-point arithmetic, round-off error is almost produced in each addition, subtraction or multiplication in Clenshaw algorithm. We choose the recurrence relation at ith step of Clenshaw Algorithm for analysis, we have

$$\hat{b}_j = (A_j^{(h)} \otimes x \oplus B_j^{(h)}) \otimes b_{j+1} \oplus C_{j+1}^{(h)} \otimes b_{j+2} \oplus a_j. \tag{8}$$

As for theoretical computation related with (8), we deduce that

$$\begin{aligned} b_j &= (A_j x + B_j) b_{j+1} + C_{j+1} b_{j+2} + a_j \\ &= [(A_j^{(h)} + A_j^{(l)} + A_j^{(m)})x + (B_j^{(h)} + B_j^{(l)} + B_j^{(m)})]b_{j+1} + (C_{j+1}^{(h)} + C_{j+1}^{(l)} + C_{j+1}^{(m)})b_{j+2} + a_j. \end{aligned} \tag{9}$$

In Algorithm 4 we use EFTs and ThreeProd to eliminate every error and obtain that

$$\begin{aligned} &[s_j, \alpha_j] = \text{ThreeProd}(A_j^{(h)}, x, \hat{b}_{j+1}), \quad [t_j, \beta_j] = \text{TwoProd}(B_j^{(h)}, \hat{b}_{j+1}), \\ &[u_j, \xi_j] = \text{TwoProd}(C_{j+1}^{(h)}, \hat{b}_{j+2}), \qquad [v_j, \eta_j] = \text{TwoSum}(s_j, t_j), \\ &[w_j, \theta_j] = \text{TwoSum}(v_j, u_j), \qquad [\hat{b}_j, \zeta_j] = \text{TwoSum}(w_j, a_j). \end{aligned} \tag{10}$$

Contrast with (9), the round-off error of method (8) is

$$e_j = (A_j^{(l)} + A_j^{(m)})x\hat{b}_{j+1} + (B_j^{(l)} + B_j^{(m)})\hat{b}_{j+1} + (C_{j+1}^{(l)} + C_{j+1}^{(m)})\hat{b}_{j+2} + \tau_j + \alpha_j + \beta_j + \xi_j + \eta_j + \theta_j + \zeta_j. \tag{11}$$

Algorithm 6 Compensate Clenshaw Algorithm

function `CompClenshaw`(p, x)
Initialization the starting $\hat{b}_{n+2} = \hat{b}_{n+1} = 0, \epsilon b_{n+2} = \epsilon b_{n+1} = 0$
for $j = n : -1 : 0$ **do**
  $[A_j^{(h)}, A_j^{(l)}] = div\_d\_d(-1, n+1)$
  $[B_j^{(h)}, B_j^{(l)}] = div\_d\_d(2n+\alpha+1, n+1)$
  $[C_j^{(h)}, C_j^{(l)}] = div\_d\_d(-n-1-\alpha, n+1)$
  $[s_j, \alpha_j] = ThreeProd(A_j^{(h)}, x, \hat{b}_{j+1})$
  $[t_j, \beta_j] = TwoProd(B_j^{(h)}, \hat{b}_{j+1})$
  $[u_j, \xi_j] = TwoProd(C_{j+1}^{(h)}, \hat{b}_{j+2})$
  $[v_j, \eta_j] = TwoSum(s_j, t_j)$
  $[w_j, \theta_j] = TwoSum(v_j, u_j)$
  $[\hat{b}_j, \zeta_j] = TwoSum(w_j, a_j)$
  $\hat{s}_j = \alpha_j \oplus \beta_j \oplus \xi_j \oplus \eta_j \oplus \theta_j \oplus \zeta_j \oplus (A_j^{(l)} \otimes x \otimes \hat{b}_{j+1} \oplus B_j^{(l)} \otimes \hat{b}_{j+1} \oplus C_{j+1}^{(l)} \otimes \hat{b}_{j+2})$
  $\epsilon\hat{b}_j = A_j^{(h)} \otimes x \otimes \epsilon b_{j+1} \oplus C_{j+1}^{(h)} \otimes \epsilon\hat{b}_{j+2} \oplus \hat{s}_j$
end for
$res = \hat{b}_0 + \epsilon\hat{b}_0$

Then we can obtain Theorem 1 as follow.

Theorem 1. *Let* $p(x) = \sum_{i=0}^{n} a_k L_k^{(\alpha)}(x) p(x) = \sum_{i=0}^{n} a_i L_i^{(\alpha)}(x)$, $\hat{b}_0$ *be the computed result by Clenshaw algorithm in floating-point arithmetic, given* $e_i = \tau_i + \alpha_i + \beta_i + \xi_i + \eta_i + \theta_i + \zeta_i, i = 0, \ldots, n-1$, *we can get the error bound as*

$$\sum_{j=0}^{n-1} e_j L_j^{(\alpha)}(x) + \hat{b}_0 = \sum_{j=0}^{n} a_j L_j^{(\alpha)}(x). \tag{12}$$

As we can see, in Theorem 1, the computed result $\hat{b}_0$ is an exact value with perturbation $\sum_{j=0}^{n-1} e_j L_j^{(\alpha)}(x)$. Assuming $c = \sum_{j=0}^{n-1} e_j L_j^{(\alpha)}(x)$ and $b_0 = \sum_{j=0}^{n} a_j L_j^{(\alpha)}(x)$, we obtain

$$\hat{b}_0 + c = b_0.$$

Therefore, we record the round-off error in each step of Clenshaw algorithm by EFTs, and then compute the perturbation to correct the numerical result $\hat{b}_0$. In fact, because c is also a Laguerre series, we can apply Clenshaw algorithm again. We present compensated Clenshaw algorithm (CompClenshaw) in Algorithm 6.

In order to analyze the error bound of the CompClenshaw algorithm, we show Lemma 1 as follow.

Lemma 1. *Let $p(x) = \sum_{i=0}^{n} a_i L_i^{(\alpha)}(x)$ be a Laguerre series of degree n, x is a floating-point number, $\hat{b}_0$ the numerical result of Clenshaw algorithm. Then*

$$|\, Clenshaw(p, x) - p(x) \,| \le \gamma_{6n-2} \sum_{j=0}^{n} |\, a_j \,|\, \tilde{L}_j^{(\alpha)}(|\, x \,|). \tag{13}$$

Lemma 2. *Suppose $p(x) = \sum_{i=0}^{n} a_i L_i^{(\alpha)}(x)$ be a Laguerre series of degree n and x is a floating-point number, we assume that $\sigma_i \le \omega_i(|A_i||x||\hat{b}_{i+1}| + |B_i||\hat{b}_{i+1}| + |C_{i+1}||\hat{b}_{i+2}| + |a_i|)$,where ω_i a real numbers. Then*

$$\sum_{j=0}^{n-1} \sigma_j \tilde{L}_j^{(\alpha)}(|\, x \,|) \le n\omega_j (1 + \gamma_{6(n-1)}) \sum_{j=1}^{n} |\, a_j \,|\, \tilde{L}_j^{(\alpha)}(|\, x \,|). \tag{14}$$

Based on Lemma 2, we can obtain Lemma 3 as follow.

Lemma 3. *Suppose $g_i^{(l)} = A_i^{(l)} x \hat{b}_{i+1} + B_i^{(l)} \hat{b}_{i+1} + C_{i+1}^{(l)} \hat{b}_{i+2}, g_i^{(m)} = A_i^{(m)} x \hat{b}_{i+1} + B_i^{(m)} \hat{b}_{i+1} + C_{i+1}^{(m)} \hat{b}_{i+2}$, $s_i = \alpha_i + \beta_i + \xi_i + \eta_i + \theta_i + \zeta_i, \bar{s}_i = |\alpha_i| + |\beta_i| + |\xi_i| + |\eta_i| + |\theta_i| + |\zeta_i|$ and τ_i is the error of ThreeProd, then we can obtain*

$$|\sum_{j=0}^{n} g_j^{(l)} L_j^{(\alpha)}(x)| \le \gamma_{6n+1} \sum_{j=0}^{n} |\, a_j \,|\, \tilde{L}_j^{(\alpha)}(|\, x \,|), \tag{15}$$

$$|\sum_{j=0}^{n} g_j^{(m)} L_j^{(\alpha)}(x)| \le u\gamma_{6n+1} \sum_{j=0}^{n} |\, a_j \,|\, \tilde{L}_j^{(\alpha)}(|\, x \,|), \tag{16}$$

$$|\sum_{j=0}^{n-1} \bar{s}_j \tilde{L}_j^{(\alpha)}(x)| \le \gamma_{6n-1} \sum_{j=0}^{n} |\, a_j \,||\, \tilde{L}_j^{(\alpha)}(|\, x \,|), \tag{17}$$

$$|\sum_{j=0}^{n-1} |\, \tau_j \,|\, \tilde{L}_j^{(\alpha)}(x)| \le \gamma_6 \gamma_{6n-4} \sum_{j=0}^{n} |\, a_j \,|\, \tilde{L}_j^{(\alpha)}(|\, x \,|). \tag{18}$$

As analyzing in Lemma 3, we discover some perturbation do not influence the numerical result in working precision. Such as (16),(18), the bound $u\gamma_{6n+1}, \gamma_6\gamma_{6n-4}$ are $O(u^2)$, so these errors do not affect the precision. However, from (15),(17) we know that $\gamma_{6n+1}, \gamma_{6n-1}$ are $O(u)$, so the coefficient perturbations may influence the accuracy, we need to consider it in our error analysis.

Theorem 2. *Let $p(x) = \sum_{i=0}^{n} a_i L_i^{\alpha}(x)$ be a Laguerre series of degree $n, (n \ge 2)$ and x is a floating-point number. Then the forward error bound of the CompClenshaw algorithm is*

$$|CompClenshaw(p,x)-p(x)|\le u\,|p(x)|+2\gamma_{6n}^2\sum_{j=0}^{n}|a_j|\tilde{L}_j^{(\alpha)}(|x|). \tag{19}$$

Proof. From [9, 15], we show two inequations to get the error bound.
Suppose

$$s_j=\alpha_j+\beta_j+\xi_j+\eta_j+\theta_j+\zeta_j,$$

then we have

$$\hat{s}_j\le(1+\gamma_5)(|\alpha_j|+|\beta_j|+|\xi_j|+|\eta_j|+|\theta_j|+|\zeta_j|).$$

Considering that

$$\begin{aligned}
&|\sum_{j=0}^{n-1}s_jL_j^{(\alpha)}(x)-\oplus\sum_{j=0}^{n-1}\hat{s}_j\otimes L_j^{(\alpha)}(x)|\\
&\le|\sum_{j=0}^{n-1}s_jL_j^{(\alpha)}(x)-\sum_{j=0}^{n-1}\hat{s}_jL_j^{(\alpha)}(x)|+|\sum_{j=0}^{n-1}\hat{s}_jL_j^{(\alpha)}(x)-\oplus\sum_{j=0}^{n-1}\hat{s}_j\otimes L_j^{(\alpha)}(x)|\\
&\le\sum_{j=0}^{n-1}|s_j-\hat{s}_j|L_j^{(\alpha)}(|x|)+\gamma_{6n-7}\sum_{j=0}^{n-1}|\hat{s}_j|\tilde{L}_j^{(\alpha)}(|x|)\\
&\le\gamma_{6n-2}\sum_{j=0}^{n-1}(|\alpha_j|+|\beta_j|+|\xi_j|+|\eta_j|+|\theta_j|+|\zeta_j|)\tilde{L}_j^{(\alpha)}(|x|)\\
&\le\gamma_{6n-2}\gamma_{6n-1}\sum_{j=0}^{n}|a_j|\tilde{L}_j^{(\alpha)}(|x|)\qquad\text{[By inequation (17)]}\\
&\le\gamma_{6n-1}^2\sum_{j=0}^{n}|a_j|L_j^{(\alpha)}(|x|)
\end{aligned}$$

and

$$\begin{aligned}
&|\sum_{j=0}^{n-1}g_j^{(l)}L_j^{(\alpha)}(x)-\oplus\sum_{j=0}^{n-1}\hat{g}_j^{(l)}\otimes L_j^{(\alpha)}(x)|\\
&\le|\sum_{j=0}^{n-1}g_j^{(l)}L_j^{(\alpha)}(x)-\sum_{j=0}^{n-1}\hat{g}_j^{(l)}L_j^{(\alpha)}(x)|+|\sum_{j=0}^{n-1}\hat{g}_j^{(l)}L_j^{(\alpha)}(x)-\oplus\sum_{j=0}^{n-1}\hat{g}_j^{(l)}\otimes L_j^{(\alpha)}(x)|\\
&\le\sum_{j=0}^{n-1}|g_j^{(l)}-\hat{g}_j^{(l)}|\tilde{L}_j^{(\alpha)}(|x|)+\gamma_{6n-7}\sum_{j=0}^{n-1}|\hat{g}_j^{(l)}|\tilde{L}_j^{(\alpha)}(|x|)\\
&\le\gamma_{6n-2}\sum_{j=0}^{n-1}(|A_j^{(l)}x\hat{b}_{j+1}|+|B_j^{(l)}\hat{b}_{j+1}|+|C_{j+1}^{(l)}\hat{b}_{j+2}|),
\end{aligned}$$

combining with

$$|A_j^{(l)}x\hat{b}_{j+1}|+|B_j^{(l)}\hat{b}_{j+1}|+|C_{j+1}^{(l)}\hat{b}_{j+2}|\le u(1+\gamma_1)|A_jx\hat{b}_{j+1}|+|B_j\hat{b}_{j+1}|+|C_{j+1}\hat{b}_{j+2}|,$$

we have

$$|\sum_{j=0}^{n-1} g_j^{(l)} L_j^{(\alpha)}(x) - \oplus\sum_{j=0}^{n-1} \hat{g}_j^{(l)} \otimes L_j^{(\alpha)}(x)|$$

$$\le \gamma_{6n-2} u(1+\gamma_1) n(1+\gamma_{6(n-1)}) \sum_{j=0}^{n} |a_j| \tilde{L}_j^{(\alpha)}(|x|)$$

$$\le \gamma_{6n-2}\gamma_{6n} \sum_{j=0}^{n} |a_j| \tilde{L}_j^{(\alpha)}(|x|)$$

$$\le \gamma_{6n-1}^2 \sum_{j=0}^{n} |a_j| \tilde{L}_j^{(\alpha)}(|x|).$$

Then we obtain the error bound

$$|CompClenshaw(p,x) - p(x)| = |(1+\varepsilon)(\hat{b}_0 + \varepsilon\hat{b}_0) - p(x)|$$

$$= |(1+\varepsilon)(p(x) - \varepsilon b_0 + \varepsilon\hat{b}_0) - p(x)| \qquad \text{[By Theroem 3.1]}$$

$$\le u\,|p(x)| + (1+u)\,|\varepsilon b_0 - \varepsilon\hat{b}_0|.$$

Next we consider that

$$|\varepsilon b_0 - \varepsilon\hat{b}_0| = |\sum_{j=0}^{n-1} s_j L_j^{(\alpha)}(x) + \sum_{j=0}^{n-1} g_j^{(l)} L_j^{(\alpha)} L_j^{(\alpha)}(x) - \oplus\sum_{j=0}^{n-1} \hat{s}_j \otimes L_j^{(\alpha)}(x) - \oplus\sum_{j=0}^{n-1} \hat{g}_j^{(l)} \otimes L_j^{(\alpha)}(x)|$$

$$\le |\sum_{j=0}^{n-1} s_j L_j^{(\alpha)}(x) - \oplus\sum_{j=0}^{n-1} \hat{s}_j \otimes L_j^{(\alpha)}| + |\sum_{j=0}^{n-1} g_j^{(l)} L_j^{(\alpha)}(x) - \oplus\sum_{j=0}^{n-1} g_j^{(l)} \otimes L_j^{(\alpha)}(x)|$$

$$\le 2\gamma_{6n-1}^2 \sum_{j=0}^{n-1} |a_j| \tilde{L}_j^{(\alpha)}(|x|).$$

Finally, we obtain

$$|CompClenshaw(p,x) - p(x)| \le u\,|p(x)| + 2(1+u)\gamma_{6n-1}^2 \sum_{j=0}^{n} |a_j| \tilde{L}_j^{(\alpha)}(|x|)$$

$$\le u\,|p(x)| + 2\gamma_{6n}^2 \sum_{j=0}^{n} |a_j| \tilde{L}_j^{(\alpha)}(|x|).$$

Combining Definition 2 and Theorem 2, we present corollary 3.1 as follow. Given $p(x) = \sum_{i=0}^{n} a_i L_i^{\alpha}$ a Laguerre series of degree n with floating-point coefficients, and x is a floating-point number. Then the error bound of CompClenshaw algorithm is

$$\frac{|CompClenshaw(p,x) - p(x)|}{|p(x)|} \le u + 2\gamma_{6n}^2 cond(p,x). \tag{20}$$

As the Corollary 3.1 show, the relative error bound produced by CompClenshaw algorithm is γ_{6n}^2 times the condition number of the polynomial and plus an unavoidable summand u. That is to say, if $cond(p,x) \le 0.5\gamma_{6n}^{-2} \times O(u)$, the computed result of CompClenshaw algorithm is accurate. When $cond(p,x) \ge 0.5\gamma_{6n}^{-2} \times O(u)$, its accuracy will be the same as computed in original Clenshaw algorithm with normal condition numbers.

4. Experimental Results

4.1. Evaluation of the Polynomial in Laguerre basis

All our experiments are performed using IEEE-754 double precision. The polynomials in Laguerre basis we consider are floating-point coefficients, and x is a floating point. We use the Symbolic Toolbox in Matlab to obtain exact results for comparisons.

In our experiment, we consider polynomial $p(x) = (x-0.52)^7(x-1)^{10}$, an ill-conditional polynomial in neighborhood of its multiple roots 0.52 and 1. The Clenshaw and CompClenshaw Algorithm are used to evaluate the Laguerre polynomial basis with parameter $\alpha = 0$. For perturbations of Laguerre polynomial, we need to convert the polynomial coefficients to Laguerre basis [17]. We use a Convert algorithm [15] to obtain the coefficients of Laguerre polynomial.

Firstly, we check the polynomial with argument $\alpha = 0$ for 400 isometric points in [0.68,1.15], [0.7485, 0.7515], and [0.993,1.007]. We show the absolute error (13) and (19) in Figure 1. From Figure 1, we detect that the CompClenshaw algorithm is more accurate than Clenshaw.

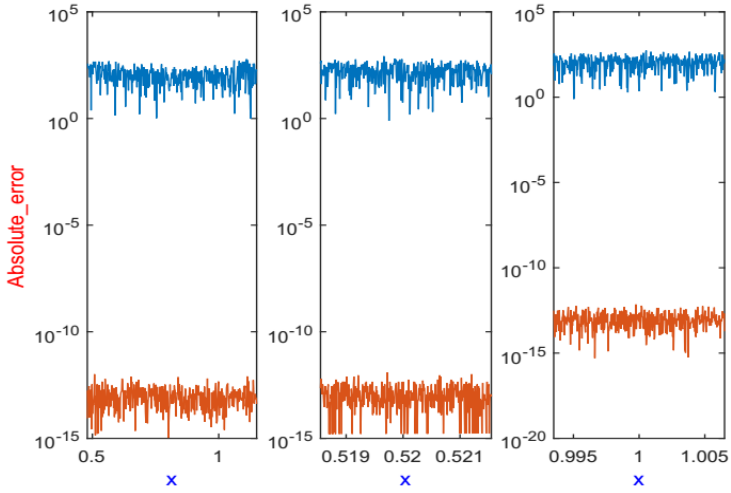

Fig. 1. Absolute error of polynormial from $p(x) = (x-0.75)^7(x-1)^{10}$ by the Convert algorithm in Laguerre basis in neighborhood of its multiple roots.using (13) [upper line], the (19) [downward line].

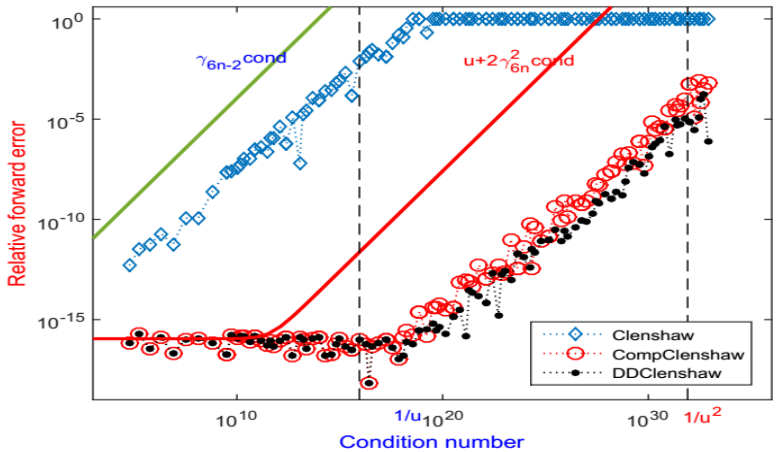

Fig. 2. Forward error of polynomial $p(x) = (x-0.52)^7(x-1)^{10}$ evaluate by `Clenshaw`, `CompClenshaw` and `DDClenshaw` algorithm.

As we known, the closer to the root, the larger the condition number [10]. We test 120 points of polynomial $p(x) = (x-0.52)^7(x-1)^{10}$ near the root $x = 0.52$, i.e. $x = 0.52 - 1.03^{2i-85}$ where $i = 1:80$ and $x = 1 - 1.03^{5i-85}$ where $i = 1:40$. We compare the forward relative error $|p_{res}(x) - p_{sym}(x)|/p_{sym}(x)$ of Clenshaw, CompClenshaw and DDClenshaw algorithm in Figure 2, respectively. As we can see, in Figure 2, the relative error of CompClenshaw and DDClenshaw algorithm are almost same. Moreover, when condition number is less than working precision u ($u = 1.16 \times 10^{-16}$), the result of

CompClenshaw is nearly equal to u. When condition number is larger than u, the forward error increase linearly.

Meanwhile, we also consider the computational complexity of Clenshaw, CompClenshaw, and DDClenshaw algorithms, based on the computation cost of TwoSum, TwoProd, and FastTwoSum. The complexity of them is $7n-2$ flops, $105n-9$ flops, $154n-44$ flops . So we can observe that CompClenshaw is as accurate as DDClenshaw in double precision, but it only needs about 68.18% of flops.

5. Conclusion

This paper introduce a CompClenshaw algorithm to evaluate the Laguerre polynomial. The Clenshaw algorithm is not accurate enough when the problem is ill-conditioned, particularly the floating-point is near the multiple root of polynomial. The CompClenshaw algorithm can delay the appearance of instability problems in standard precision and obtain a full precision result, which as same as computed in double-double precision. Besides, the CompClenshaw algorithm is more efficient to evaluate Laguerre polynomial.

Acknowledgements

Partially supported by National Natural Science Foundation of China (No. 61571008). Partially supported by National Natural Science Foundation of China (No. 61402495, No. 61602166, No. 61303189, No. 61402496). Partially supported by Science Project of National University of Defense Technology (JC120201) and National Natural Science Foundation of Hunan Province in China (13JJ2001).

References

[1] C. Hwang and Y. P. Shih. Parameter identification via laguerre polynomials. International Journal of Systems Science, 13(2):209–217, 1982.

[2] M. Gülsu, B. Gürbüz, Y. Öztürk, and M. Sezer. Laguerre polynomial approach for solving linear delay difference equations. *Applied Mathematics and Computation*, 217(15):6765–6776, 2011.

[3] R. Piessens. Numerical inversion of the laplace transform using generalised laguerre polynomials. *Proceedings of the Institution of Electrical Engineers*, 118(10):1517–1522, 1971. *New York*, 19852:309, 1985.

[4] M. W. Coffey. Generalized raising and lowering operators for supersymmetric quantum mechanics. *Physics*, 2015.

[5] C. W. Clenshaw. A note on the summation of chebyshev series. *Mathematics of Computation*, 9(51):118–120, 1955.

[6] A. Smoktunowicz. Backward stability of clenshaw's algorithm. *BIT Numerical Mathematics*, 42(3):600–610, 2002.

[7] R. Barrio. Rounding error bounds for the clenshaw and forsythe algorithms for the evaluation of orthogonal polynomial series. *Journal of computational and applied mathematics*, 138(2):185–204, 2002.

[8] H. Jiang, R. Barrio, H. Li, X. Liao, L. Cheng, and F. Su. Accurate evaluation of a polynomial in chebyshev form. *Applied Mathematics and Computation*, 217(23):9702–9716, 2011.

[9] H. Jiang, S. Graillat, C. Hu, S. Li, X. Liao, L. Cheng, and F. Su. Accurate evaluation of the k-th derivative of a polynomial and its application. *Journal of Computational and Applied Mathematics*, 243:28–47, 2013.

[10] P. Langlois, S. Graillat, and N. Louvet. Compensated horner scheme. 2005.

[11] T. Ogita, S. M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6):1955–1988, 2005.

[12] IEEE. IEEE standard for binary floating-point arithmetic. 1985.

[13] D. E. Knuth. *The art of computer programming*. Addison-Wesley Pub. Co., 1973. [13] D. E. Knuth. *The art of computer programming*. Addison-Wesley Pub. Co., 1973.

[14] T. J. Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, 1970.

[15] P. Du, H. Jiang, and L. Cheng. Accurate evaluation of polynomials in legendre basis. *Journal of Applied Mathematics*, 2014(1):1–13, 2014.

[16] R. Barrio H. Jiang and S. Serrano. A general condition number for polynomials. *SIAM Journal on Numerical Analysis*, 51(2):1280–1294, 2013.
[17] R. Barrio and J. M. Peña. Basis conversions among univariate polynomial representations. *Comptes Rendus Mathematique*, 339(4):293–298, 2004.
[18] S. R. Graillat. Accurate floating-point product and exponentiation. *IEEE Transactions on Computers*, 58(7):994–1000, 2009.
[19] D. H. Bailey High-precision software directory: QD library, double-double library. *Lawrence Berkeley National Laboratory*, http://www. nersc. gov/ dhbailey/ mpdist/ mpdist. html. , 2008.

Appendix

Appendix A: The Extra Proof

Proof of Theorem 1. Inspired by [15], we set

$$s_j = \tau_j + \alpha_j + \beta_j + \xi_j + \eta_j + \theta_j + \zeta_j,$$
$$t_j = (A_j^{(l)} + A_j^{(m)})x\hat{b}_{j+1} + (B_j^{(l)} + B_j^{(m)})\hat{b}_{j+1} + (C_{j+1}^{(l)} + C_{j+1}^{(m)})\hat{b}_{j+2}.$$

Then

$$\begin{aligned}\hat{b}_j + e_j &= (\hat{b}_j + s_j) + t_j \\ &= (A_j^{(h)}x + B_j^{(h)})\hat{b}_{j+1} + C_{j+1}^{(h)}\hat{b}_{j+2} + a_j + t_j \\ &= A_j x\hat{b}_{j+1} + B_j\hat{b}_{j+1} + C_{j+1}\hat{b}_{j+2} + a_j, \quad j = 1,...,n,\end{aligned} \tag{21}$$

and $\hat{b}_0 + e_0 = -x\hat{b}_1 + \alpha\hat{b}_1 + \alpha\hat{b}_2 + a_0$.

Thus

$$\hat{b}_0 = \sum_{j=0}^{n}(a_j - e_j)L_j^{(\alpha)}(x), \tag{22}$$

with $\alpha_n = \beta_n = \xi_n = \eta_n = \theta_n = \zeta_n = 0$, then we can obtain the result.

Proof of Lemma 1. Based on equation (3), we obtain

$$\begin{aligned}\hat{b}_{n-1} &= A_{n-1}^{(h)} \otimes x \otimes \hat{b}_n \oplus B_{n-1}^{(h)} \otimes \hat{b}_n \oplus \hat{a}_j^{(h)} \\ &= A_{n-1}x\hat{b}_n < 5 > + B_{n-1}\hat{b}_n < 4 > + a_{n-1} < 1 > \\ \hat{b}_{n-2} &= [A_{n-2}^{(h)} \otimes x \oplus B_{n-2}^{(h)}] \otimes \hat{b}_{n-1} \oplus C_{n-1}^{(h)} \otimes \hat{b}_n \oplus a_{n-2}^{(h)} \\ &= A_{n-2}x\hat{b}_{n-1} < 6 > + B_{n-2}\hat{b}_{n-1} < 5 > + C_{n-1}\hat{b}_n < 4 > + a_{n-2} < 1 > \\ &\vdots\end{aligned} \tag{23}$$

Using mathmematical induction, we can obtain $\hat{b}_0$ as follow

$$\begin{aligned}\hat{b}_0 &= < 10 + 6(n-2) > (\prod_{j=0}^{n-1}A_j)x^n a_n + \cdots + a_0 < 1 > \\ &= < 6n - 2 > (\prod_{j=0}^{n-1}A_j)x^n a_n + \cdots + a_0 < 1 >.\end{aligned} \tag{24}$$

Combing (24) the forward error is

$$|\hat{b}_0 - p(x)| = |\theta_{6n-2}(\prod_{j=0}^{n-1} A_j)x^n a_n + \cdots + a_0\theta_1|$$

$$\leq \gamma_{6n-2}|(\prod_{j=0}^{n-1} A_j)||x^n||a_n| + \cdots + |a_0| \tag{25}$$

$$= \gamma_{6n-2}\sum_{j=0}^{n}|a_j|\tilde{L}_j^{(\alpha)}(|x|).$$

Proof of Lemma 2. According to (23), we have

$$|\hat{b}_j| \leq (1+\gamma_6)(|A_j||x||\hat{b}_{j+1}| + |B_j||\hat{b}_{j+1}| + |C_{j+1}||\hat{b}_{j+2}| + |a_j|). \tag{26}$$

Combining (26) with $\sigma_j \leq \omega_j(A_j|x||\hat{b}_{j+1}| + |B_j||\hat{b}_{j+1}| + |C_{j+1}||\hat{b}_{j+2}| + |a_j|)$, we get

$$\sigma_j \leq \omega_j(1+\gamma_{6(n-j-1)})\sum_{k=j}^{n}|a_k|\tilde{L}_{k-j}^{(\alpha)}(|x|), \tag{27}$$

then

$$\sum_{j=0}^{n-1}\sigma_j\tilde{L}_j^{(\alpha)}(|x|) \leq \omega_j(1+\gamma_{6(n-1)}) \times \sum_{j=0}^{n-1}\left(\sum_{k=j}^{n}|a_k|\tilde{L}_{k-j}^{(\alpha)}(|x|)\right)\tilde{L}_j^{(\alpha)}(|x|). \tag{28}$$

Then we can obtain

$$\sum_{j=0}^{n-1}\sigma_j\tilde{L}_j^{(\alpha)}(|x|) \leq \omega_j(1+\gamma_{6(n-1)})\sum_{j=0}^{n-1}\sum_{k=j}^{n}|a_k|\tilde{L}_k^{(\alpha)}(|x|) \leq n\omega_j(1+\gamma_{6(n-1)}\sum_{j=0}^{n-1}|a_j|\tilde{L}_j^{(\alpha)}(|x|)). \tag{29}$$

Proof of Lemma 3. According to $A^{(l)} \leq uA^{(h)}$ and $A^{(h)} = A < 1 >$,we obtain

$$|g_j^{(l)}| \leq u(1+\gamma_1)(|A_j||x||\hat{b}_{j+1}| + |B_j||\hat{b}_{j+1}| + |C_{j+1}||\hat{b}_{j+2}|).$$

Then combing Lemma 2

$$|\sum_{j=0}^{n} g_j^{(l)}L_j^{(\alpha)}(x)| \leq (n+1)u(1+\gamma_{6n+1})\sum_{j=0}^{n}|a_j|\tilde{L}_j^{(\alpha)}(|x|)$$

$$\leq (6n+1)u(1+\gamma_{6n+1})\sum_{j=0}^{n}|a_j|\tilde{L}_j^{(\alpha)}(|x|)$$

$$\leq \gamma_{6n+1}\sum_{j=0}^{n}|a_j|\tilde{L}_j^{(\alpha)}(|x|).$$

Similarly, we have

$$|g_j^{(m)}| \leq u^2(1+\gamma_1)(|A_j||x||\hat{b}_{j+1} + |B_j||x||\hat{b}_{j+1}| + |C_{j+1}||\hat{b}_{j+2}|).$$

Then

$$|\sum_{j=0}^{n} g_j^{(m)}L_j^{(\alpha)}(x)| \leq u\gamma_{6n+1}\sum_{j=0}^{n}|a_j|\tilde{L}_j^{(\alpha)}(|x|)$$

Based on equation (4), we obtain

$$|\alpha_j| \le (2u+14u^2)(1+\gamma_1)|A_j x\hat{b}_{j+1}|,$$
$$|\beta_j| \le u|B_j^{(h)}\hat{b}_{j+1}| \le u(1+\gamma_1)|B_j\hat{b}_{j+1}|,$$
$$|\xi_j| \le u|C_{j+1}^{(h)}\hat{b}_{j+2}| \le u(1+\gamma_1)|C_{j+1}\hat{b}_{j+2}|,$$
$$|\eta_j| \le u|s_j+t_j| \le u|A_j x\hat{b}_{j+1} < 3 > + B_j\hat{b}_{j+1} < 2 >|,$$
$$|\theta_j| \le u|A_j x\hat{b}_{j+1} < 4 > + B_j\hat{b}_{j+1} < 3 > + C_{j+1}\hat{b}_{j+2} < 2 >|,$$
$$|\zeta_j| \le u|A_j x\hat{b}_{j+1} < 5 > + B_j\hat{b}_{j+1} < 4 > + C_{j+1}\hat{b}_{j+2} < 3 > + a_j < 1 >|,$$

Then

$$\bar{s}_j \le (5u+14u^2)|1+\gamma_5| \times (|A_j x||\hat{b}_{j+1}| + |B_j\hat{b}_{j+1}| + |C_{j+1}\hat{b}_{j+2}| + |a_j|). \tag{30}$$

Let $\omega_j = (5u+14u^2)|1+\gamma_5|$, from Lemma 2 we have

$$\begin{aligned}
&|\sum_{j=0}^{n-1}\left(|\alpha_j| + |\beta_j| + |\xi_j| + |\eta_j| + |\theta_j| + |\zeta_j|\right)L_j^{(\alpha)}(x)| \\
&\le (5+14u)nu(1+\gamma_{6n-1})\sum_{j=0}^{n}|a_j||\tilde{L}_j^{(\alpha)}(|x|) \\
&\le \gamma_{6n-1}\sum_{j=0}^{n}|a_j||\tilde{L}_j^{(\alpha)}(|x|), \qquad (n>1).
\end{aligned}$$

Similarly, combing with $|\tau_j| \le \gamma_2\gamma_6|A_j^{(h)}x\hat{b}_{j+1}| \le \gamma_2\gamma_6(1+\gamma_1)|A_j x\hat{b}_{j+1}|$, we have

$$\begin{aligned}
&|\sum_{j=0}^{n-1}|\tau_j||L_j^{(\alpha)}(x)| \le n\gamma_2\gamma_6(1+\gamma_1)(1+\gamma_{6(n-1)})\sum_{j=0}^{n}|a_j||L_j^{(\alpha)}(|x|) \\
&\le \gamma_6\gamma_{6n-4}\sum_{j=0}^{n}|a_j||\tilde{L}_j^{(\alpha)}(|x|).
\end{aligned}$$

Appendix B: The double-double Algorithm [19]

Algorithm 7 Addition of double-double number and a double number

$[rh, rl] = add\_dd\_d(ah, al, b)$
$[th, tl] = TwoSum(ah, b)$
$tl = al \oplus tl$
$[rh, rl] = FastTwoSum(th, tl)$

Algorithm 8 Multiplication of a double-double number by a double number

$[rh, rl] = prod\_dd\_d(ah, al, b)$
$[th, tl] = TwoProd(ah, b)$
$tl = al \otimes b \oplus tl$
$[rh, rl] = FastTwoSum(th, tl)$

Algorithm 9 Multiplication of a double-double number by a double-double number

$[rh, rl] = prod\_dd\_dd(ah, al, bh, bl)$
$[th, tl] = TwoProd(ah, bh)$
$tl = (ah \otimes bl) \oplus (al \otimes bh) \oplus tl$
$[rh, rl] = FastTwoSum(th, tl)$

Algorithm 10 Addition of a double-double number and a double-double number

$[rh, rl] = add\_dd\_dd(ah, al, bh, bl)$

$[sh, sl] = TwoSum(ah, bh)$

$[th, tl] = TwoSum(al, bl)$

$sl = sl \oplus th$

$th = sh \oplus sl$

$sl = sl \ominus (th \ominus sh)$

$tl = tl \oplus sl$

$[rh, rl] = FastTwoSum(th, tl)$

Algorithm 11 DDCleshaw algorithm of evaluating Laguerre series in double-double arthmetic

function $res = DDClenshaw(p, x)$

$b_{n+2} = b_{n+1} = 0$

for $j = n : -1 : 0$

$[rh1, rl1] = prod\_dd\_d(b_{j+1}^{(h)}, b_{j+1}^{(h)}, x)$

$[rh2, rl2] = prod\_dd\_dd(rh1, rl1, A_j^{(h)}, A_j^{(l)}))$

$[rh3, rl3] = prod\_dd\_dd(b_{j+1}^{(h)}, b_{j+1}^{(l)}, B_j^{(h)}, B_j^{(l)})$

$[rh4, rl4] = prod\_dd\_dd(b_{j+2}^{(h)}, b_{j+2}^{(l)}, C_{j+1}^{(h)}, C_{j+1}^{(l)})$

$[rh5, rl5] = add\_dd\_dd(rh2, rl2, rh3, rl3)$

$[rh6, rl6] = add\_dd\_dd(rh5, rl5, -rh4, -rl4)$

$[b_j^{(h)}, b_j^{(l)}] = add\_dd\_d(rh6, rl6, a_j)$

end for

$res = [b_0^{(h)}, b_0^{(l)}]$