Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 742538, 13 pages
http://dx.doi.org/10.1155/2014/742538

Research Article

Accurate Evaluation of Polynomials in Legendre Basis

Peibing Du,[1] Hao Jiang,[2] and Lizhi Cheng[1]

[1] *Department of Mathematics and Systems Science, College of Science, National University of Defense Technology, Changsha 410073, China*
[2] *The State Key Laboratory for High Performance Computation, College of Computer Science, National University of Defense Technology, Changsha 410073, China*

Correspondence should be addressed to Peibing Du; duli_0211@foxmail.com

Received 22 April 2014; Accepted 2 July 2014; Published 23 July 2014

Academic Editor: Roberto Barrio

This paper presents a compensated algorithm for accurate evaluation of a polynomial in Legendre basis. Since the coefficients of the evaluated polynomial are fractions, we propose to store these coefficients in two floating point numbers, such as double-double format, to reduce the effect of the coefficients' perturbation. The proposed algorithm is obtained by applying error-free transformation to improve the Clenshaw algorithm. It can yield a full working precision accuracy for the ill-conditioned polynomial evaluation. Forward error analysis and numerical experiments illustrate the accuracy and efficiency of the algorithm.

1. Introduction

Legendre polynomial is often used in numerical analysis [1–3], such as approximation theory and quadrature and differential equations. Legendre polynomial satisfies 3-term recurrence relation; that is, for Legendre polynomial $p_k(x)$,

$$\begin{aligned} p_0(x) &= 1 \\ p_1(x) &= x \\ p_{k+1}(x) &= \frac{2k+1}{k+1} x p_k(x) - \frac{k}{k+1} p_{k-1}(x) \quad (k > 0). \end{aligned} \tag{1}$$

The polynomial represented in Legendre basis is $p(x) = \sum_{j=0}^{n} a_j p_j(x)$, where $a_j \in \mathbb{R}$ and $p_j(x)$ is Legendre polynomial.

The Clenshaw algorithm [4, 5] is usually used to evaluate a linear combination of Chebyshev polynomials, but it can apply to any class of functions that can be defined by a three-term recurrence relation. Therefore the Clenshaw algorithm can evaluate a polynomial in Legendre basis. The error analysis of the Clenshaw algorithm was considered in the literatures [6–10]. The relative accuracy bound of the computed values $\widehat{p}(x)$ by the Clenshaw algorithm verifies

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

For ill-conditioned problems, several researches applied error-free transformations [11] to propose accurate compensated algorithms [12–15] to evaluate the polynomials in monomial, Bernstein, and Chebyshev bases with Horner, de Casteljau, and Clenshaw algorithms, respectively. Some recent applications of high-precision arithmetic were given in [16].

Motivated by them, we apply error-free transformations to analyze the effect of round-off errors and then compensate them to the original result of the Clenshaw algorithm. Since the coefficients of the Legendre polynomial are fractions, the coefficient perturbations in the evaluation may exist when the coefficients are truncated to floating point numbers. We store the coefficients which are not floating point numbers in double-double format, with the double working precision, to get the perturbation. We also compensate the approximate perturbed errors to the original result of the Clenshaw algorithm. Based on the above, we construct a compensated Clenshaw algorithm for the evaluation of a linear combination of Legendre polynomials, which can yield a full working precision accuracy and its relative accuracy bound satisfies

$$\frac{|p(x)-\widehat{p}(x)|}{p(x)} \leq o(u) + \operatorname{cond}(p,x) \times o(u^2). \tag{3}$$

The paper is organized as follows. Section 2 shows some basic notations in error analysis, floating point arithmetic, error-free transformations, compensated algorithm, Clenshaw algorithm, and condition number. Section 3 presents the compensated algorithm and its error bound. Section 4 gives several numerical experiments to illustrate the efficiency and accuracy of the compensated algorithm for polynomial in Legendre basis.

2. Mathematical and Arithmetical Preliminaries

2.1. Basic Notations and Definitions. Throughout this paper, we assume to work with a floating point arithmetic adhering to IEEE-754 floating point standard in rounding to nearest and no overflow nor underflow occurs. Let $op \in \{\oplus, \ominus, \otimes, \oslash\}$ represent the floating point computation; then the computation obeys the model

$$a\, op\, b = (a \circ b)(1+\varepsilon_1) = \frac{(a \circ b)}{(1+\varepsilon_2)}, \tag{4}$$

where $\circ \in \{+, -, \times, \div\}$ and $|\varepsilon_1|, |\varepsilon_2| \leq u$ (u is the round-off unit). We also assume that the computed result of $a \in \mathbb{R}$ in floating point arithmetic is denoted by $\widehat{a}$ and the set of all floating point numbers is denoted by $\mathbb{F}$. The following definition will be used in error analysis (see more details in [17]).

Definition 1. One defines

$$\langle n \rangle := \prod_{i=1}^{n} (1+\delta_i)^{\rho_i} = 1 + \theta_n, \tag{5}$$

where $|\delta_i| \leq u$, $\rho_i = \pm 1$ with $\forall i = 1, 2, \ldots, n$, $|\theta_n| \leq \gamma_n := (nu/(1-nu)) = nu + O(u^2)$, and $nu < 1$.

There are three classic properties which will also be used in error analysis:

(i) $\langle k \rangle \langle j \rangle = \langle k+j \rangle$;

(ii) $\gamma_k < \gamma_{k+1}$;

(iii) $\gamma_k + \gamma_j + \gamma_k \gamma_j \leq \gamma_{k+j}$.

2.2. Accurate Sum and Product. Let $a, b \in \mathbb{F}$, and no overflow nor underflow occurs. The transformation $(a,b) \rightarrow (x,y)$ is regarded as an error-free transformation (EFT) that causes $y \in \mathbb{F}$ to exist such that $a \circ b = x + y$, $x = a\, op\, b$.

Let us show the error-free transformations of the sum and product of two floating point numbers in Algorithms 1-3 which are the *TwoSum* algorithm by Knuth [18] and the *TwoProd* algorithm by Dekker [19], respectively.

Algorithms 1–3 satisfy the Theorem 2.

```
function $[x, y] = TwoSum(a, b)$
$x = a \oplus b$
$z = x \ominus a$
$y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$
```

Algorithm 1: Sum of two floating point numbers.

```
function $[x, y] = Split(a)$
$c = factor \otimes a$ (in double precision factor = $2^{27} + 1$)
$x = c \ominus (c \ominus a)$
$y = a \ominus x$
```

Algorithm 2: Split of a floating point number into two parts.

```
function $[x, y] = TwoProd(a, b)$
$x = a \otimes b$
$[ah, al] = Split(a)$
$[bh, bl] = Split(b)$
$y = al \otimes bl \ominus (((x \ominus ah \otimes bh) \ominus al \otimes bh) \ominus ah \otimes bl)$
```

Algorithm 3: Product of two floating point numbers.

```
function $[x, y] = ThreeProd(a, b, c)$
$[t, h] = TwoProd(a, b)$
$[x, e] = TwoProd(t, c)$
$y = c \otimes h \oplus e$
```

Algorithm 4: Product of three floating point numbers.

Theorem 2 (see [11]). *For $a, b \in \mathbb{F}$ and $x, y \in \mathbb{F}$, TwoSum and TwoProd verify*

$$\begin{gathered} [x, y] = TwoSum(a, b), \qquad x + y = a + b, \\ |y| \leq \min\{u|x|, u|a+b|\}, \\ [x, y] = TwoProd(a, b), \qquad x + y = ab, \\ |y| \leq \min\{u|x|, u|ab|\}. \end{gathered} \tag{6}$$

We present the compensated algorithm for the product of three floating point numbers in Algorithm 4, which refers to [20].

According to Theorem 2, we have $|h| \leq u|ab|$ and $t + h = ab$. Hence, $|e| \leq u|(ab-h)c| \leq u|abc| + u|hc| \leq u|abc| + u^2|abc|$; then $|ch + e| \leq |ch| + |e| \leq 2u|abc| + u^2|abc|$. According to Lemma 3.2 in [20], we can propose the error bound of Algorithm 4 in Theorem 3.

Theorem 3. *For $a, b, c \in \mathbb{F}$ and $x, y, \tau \in \mathbb{F}$, ThreeProd verifies*

$$\begin{gathered}[x, y] = ThreeProd(a, b, c), \qquad x + y = abc - \tau, \\ |\tau| \leq \gamma_2\gamma_6 |abc|, \qquad |y| \leq 2u |abc| + 14u^2 |abc|.\end{gathered} \tag{7}$$

Proof. According to Lemma 3.2 in [20], we have

$$|\tau| \leq \gamma_2\gamma_6 |abc|. \tag{8}$$

In Algorithm 4, $y + \tau = ch + e$; then

$$\begin{aligned}|y| = |y + \tau - \tau| &\leq |ch + e| + |\tau| \\ &\leq 2u |abc| + u^2 |abc| + \gamma_2\gamma_6 |abc| \\ &\leq 2u |abc| + 14u^2 |abc|.\end{aligned} \tag{9}$$

□

2.3. The Clenshaw Algorithm and Condition Number. The standard general algorithm for the evaluation of polynomial in Legendre basis $p(x) = \sum_{j=0}^{n} a_j p_j(x)$ is the Clenshaw algorithm [5]. We recall it in Algorithm 5.

Barrio et al. [21] proposed a general polynomial condition number for any polynomial basis defined by a linear recurrence and used this new condition number to give the error bounds for the Clenshaw algorithm. Based on this general condition number, we propose the absolute Legendre polynomial, which is similar to the absolute polynomial mentioned in [7, 21].

Definition 4. Let $p_k(x)$ be Legendre polynomial. We define the absolute Legendre polynomial $\tilde{p}_k(x)$ which is associated with $p_k(x)$ and satisfies

$$\begin{gathered}\tilde{p}_0(x) = 1 \\ \tilde{p}_1(x) = x \\ \tilde{p}_k(x) = \frac{2k-1}{k} x \tilde{p}_{k-1}(x) + \frac{k}{k+1} \tilde{p}_{k-2}(x) \quad (k \geq 2),\end{gathered} \tag{10}$$

where $|p_k(x)| \leq \tilde{p}_k(x), \forall x \geq 0$.

The absolute Legendre polynomial satisfies the following property.

Lemma 5. *Let $\tilde{p}_k(x)$ be the absolute Legendre polynomial. Then we have*

$$\tilde{p}_k(x)\, \tilde{p}_j(x) \leq \tilde{p}_{k+j}(x). \tag{11}$$

Proof. Let $A_k = (2k+1)/(k+1)$ and $k \geq 2$. From Definition 4, we have

$$\begin{aligned}\tilde{p}_k(x) &= A_{k-1} x \tilde{p}_{k-1}(x) + \frac{k}{k+1} \tilde{p}_{k-2}(x) \\ &= \left(\prod_{i=0}^{k-1} A_i\right) x^k + \cdots;\end{aligned} \tag{12}$$

```
function $p(x) = Clenshaw(p, x)$
$b_{n+2} = b_{n+1} = 0$
for $j = n : -1 : 0$
    $b_j = \frac{2j+1}{j+1} x b_{j+1} - \frac{j+1}{j+2} b_{j+2} + a_j$
end
$p(x) = b_0$
```

Algorithm 5: The Clenshaw algorithm for finite Legendre series.

then

$$\tilde{p}_k(x)\, \tilde{p}_j(x) = \left(\prod_{i=0}^{k-1} A_i\right)\left(\prod_{i=0}^{j-1} A_i\right) x^{k+j} + \cdots. \tag{13}$$

Thus the equivalent form of (11) is

$$\left(\prod_{i=0}^{k-1} A_i\right)\left(\prod_{i=0}^{j-1} A_i\right) \leq \prod_{i=0}^{k+j-1} A_i. \tag{14}$$

Since A_k is increasing with k, we obtain

$$\prod_{i=0}^{j-1} A_i \leq \prod_{i=k}^{k+j-1} A_i. \tag{15}$$

So we finally obtain $\tilde{p}_k(x)\tilde{p}_j(x) \leq \tilde{p}_{k+j}(x)$. □

Following Definition 4, we introduce the condition number for the evaluation of polynomials in Legendre basis [21].

Definition 6. Let $p(x) = \sum_{k=0}^{n} a_k p_k(x)$, where $p_k(x)$ is Legendre polynomial. Let $\tilde{p}_k(x)$ be the absolute Legendre polynomial. Then the absolute condition number is

$$\operatorname{cond}_{\text{abs}}(p, x) = \tilde{p}(|x|) = \sum_{k=0}^{n} |a_k|\, \tilde{p}_k(|x|), \tag{16}$$

and the relative condition number is

$$\operatorname{cond}_{\text{rel}}(p, x) = \frac{\tilde{p}(|x|)}{|p(x)|} = \frac{\sum_{k=0}^{n} |a_k|\, \tilde{p}_k(|x|)}{|p(x)|}. \tag{17}$$

3. Compensated Algorithm for Evaluating Polynomials

In this section, we exhibit the exact round-off errors generated by the Clenshaw algorithm with EFT. We also analyze the perturbations generated by truncating the fractions in Algorithm 5. We propose a compensated Clenshaw algorithm to evaluate finite Legendre series and present its error bound in the following.

Firstly, in order to analyze the perturbations, we split each coefficient into three parts as follows:

$$A = A^{(h)} + A^{(l)} + A^{(m)}, \tag{18}$$

where $A^{(h)}, A^{(l)} \in \mathbb{F}, A, A^{(m)} \in \mathbb{R}$ and $|A^{(l)}| \le u|A^{(h)}|, |A^{(m)}| \le u|A^{(l)}|$. Let $A_k = (2k+1)/(k+1)$ and $C_k = -(k/(k+1))$. Then we describe the recurrence relation at jth step of Algorithm 5 for theoretical computation as

$$\begin{aligned} b_j = {} & \left(A_j^{(h)} + A_j^{(l)} + A_j^{(m)}\right) x b_{j+1} \\ & + \left(C_{j+1}^{(h)} + C_{j+1}^{(l)} + C_{j+1}^{(m)}\right) b_{j+2} \\ & + \left(a_j^{(h)} + a_j^{(l)} + a_j^{(m)}\right). \end{aligned} \tag{19}$$

For numerical computation associated with (19), it is

$$\widehat{b}_j = A_j^{(h)} \otimes x \otimes \widehat{b}_{j+1} \oplus C_{j+1}^{(h)} \otimes \widehat{b}_{j+2} \oplus a_j^{(h)}. \tag{20}$$

Remark 7. Let $A \in \mathbb{R}$. We split a coefficient A like (18); then the representation of splitting is unique and $\widehat{A} = A^{(h)} \oplus A^{(l)}$, $\widehat{A} = A\langle 1 \rangle$.

Let $A_j = A_j^{(h)}$ and $C_j = C_j^{(h)}$ in Algorithm 5. Since every elementary floating point operation in Algorithm 5 causes round-off errors in numerical computation, we apply *EFT* and the *ThreeProd* algorithms to take notes of all round-off errors and obtain

$$\begin{aligned} \left[r, \alpha_j\right] &= ThreeProd\left(A_j^{(h)}, x, \widehat{b}_{j+1}\right), \\ \left[s, \beta_j\right] &= TwoProd\left(C_{j+1}^{(h)}, \widehat{b}_{j+2}\right), \\ \left[t, \eta_j\right] &= TwoSum\left(r, s\right), \\ \left[\widehat{b}_j, \xi_j\right] &= TwoSum\left(t, a_j^{(h)}\right). \end{aligned} \tag{21}$$

The sum of the perturbation and the round-off errors of the recurrence relation at jth step is

$$\begin{aligned} w_j = {} & \left(A_j^{(l)} + A_j^{(m)}\right) x \widehat{b}_{j+1} + \left(C_{j+1}^{(l)} + C_{j+1}^{(m)}\right) \widehat{b}_{j+2} \\ & + \left(a_j^{(l)} + a_j^{(m)}\right) + \alpha_j + \tau_j + \beta_j + \eta_j + \xi_j, \end{aligned} \tag{22}$$

where $r + \alpha_j + \tau_j = A_j^{(h)} x \widehat{b}_{j+1}$, τ_j is defined in Theorem 3. Then we obtain the following theorem.

Theorem 8. *Let $p(x) = \sum_{j=0}^{n} a_j p_j(x)$ be a Legendre series of degree n, and let x be a floating point value. One assumes that $A_j = A_j^{(h)}$ and $C_j = C_j^{(h)}$ in Algorithm 5 and $\widehat{b}_0$ is the numerical result; w_j is described in (22) for $j = 0, 1, \ldots, n-1$. Then one obtains*

$$\sum_{j=0}^{n-1} w_j p_j(x) + \widehat{b}_0 = \sum_{j=0}^{n} a_j p_j(x). \tag{23}$$

Proof. The perturbation of Algorithm 5 is

$$\left(A_j^{(l)} + A_j^{(m)}\right) x \widehat{b}_{j+1} + \left(C_{j+1}^{(l)} + C_{j+1}^{(m)}\right) \widehat{b}_{j+2} + \left(a_j^{(l)} + a_j^{(m)}\right). \tag{24}$$

Let $A_j = A_j^{(h)}$ and $C_j = C_j^{(h)}$ in Algorithm 5. From Theorems 2–3 and (21), we can easily obtain

$$\begin{aligned} A_j^{(h)} \widehat{b}_{j+1} x &= r + \alpha_j + \tau_j, \\ C_{j+1}^{(h)} \widehat{b}_{j+2} &= s + \beta_j, \\ r + s &= t + \eta_j \\ t + a_j^{(h)} &= \widehat{b}_j + \xi_j. \end{aligned} \tag{25}$$

Let $s_j = \alpha_j + \tau_j + \beta_j + \eta_j + \xi_j$ for $j = 0, 1, \ldots, n-1$; then

$$\widehat{b}_j + s_j = A_j^{(h)} \widehat{b}_{j+1} x + C_{j+1}^{(h)} \widehat{b}_{j+2} + a_j^{(h)}, \quad j = 0, 1, \ldots, n-1. \tag{26}$$

From (18), (19), (22), and (24), we have

$$\widehat{b}_j + w_j = A_j \widehat{b}_{j+1} x + C_{j+1} \widehat{b}_{j+2} + a_j, \quad j = 0, 1, \ldots, n-1. \tag{27}$$

Thus

$$\widehat{b}_0 = \sum_{j=0}^{n} \left(a_j - w_j\right) p_j(x); \tag{28}$$

then

$$\sum_{j=0}^{n-1} w_j p_j(x) + \widehat{b}_0 = \sum_{j=0}^{n} a_j p_j(x). \tag{29}$$

□

To devise a compensated algorithm of Algorithm 5 and give its error bound, we need the following lemmas.

Lemma 9. *Let $p(x) = \sum_{j=0}^{n} a_j p_j(x)$ be a Legendre series of degree n, x a floating point value, and $\widehat{b}_0$ the numerical result of Algorithm 5. Then*

$$\left|Clenshaw(p, x) - p(x)\right| \le \gamma_{5n-1} \sum_{j=0}^{n} \left|a_j\right| \widetilde{p}_j(|x|). \tag{30}$$

Proof. Applying the standard model of floating point arithmetic, from Algorithm 5 and Remark 7, we have

$$\begin{aligned} \widehat{b}_{n-1} &= A_{n-1} \widehat{b}_n x \langle 4 \rangle + a_{n-1} \langle 1 \rangle, \\ \widehat{b}_{n-2} &= A_{n-2} \widehat{b}_{n-1} x \langle 5 \rangle + C_{n-1} \widehat{b}_n \langle 4 \rangle + a_{n-2} \langle 1 \rangle, \\ &\vdots \\ \widehat{b}_0 &= A_0 \widehat{b}_1 x \langle 5 \rangle + C_1 \widehat{b}_2 \langle 4 \rangle + a_0 \langle 1 \rangle. \end{aligned} \tag{31}$$

Thus we obtain

$$\begin{aligned} \widehat{b}_0 &= \langle 4 + 5(n-1) \rangle \left(\prod_{i=0}^{n-1} A_i\right) x^n a_n + \cdots + a_0 \langle 1 \rangle \\ &= \langle 5n - 1 \rangle \left(\prod_{i=0}^{n-1} A_i\right) x^n a_n + \cdots + a_0 \langle 1 \rangle. \end{aligned} \tag{32}$$

```
function $res = CompClenshaw(p, x)$
$\widehat{b}_{n+2} = \widehat{b}_{n+1} = 0$
$\epsilon\widehat{b}_{n+2} = \epsilon\widehat{b}_{n+1} = 0$
for $j = n : -1 : 0$
    $[A_j^{(h)}, A_j^{(l)}] = div\_d\_d(2j+1, j+1)$
    $[C_j^{(h)}, C_j^{(l)}] = div\_d\_d(-j-1, j+2)$
    $[r, \alpha_j] = ThreeProd(A_j^{(h)}, \widehat{b}_{j+1}, x)$
    $[s, \beta_j] = TwoProd(C_{j+1}^{(h)}, \widehat{b}_{j+2})$
    $[t, \eta_j] = TwoSum(r, s)$
    $[\widehat{b}_j, \xi_j] = TwoSum(t, a_j)$
    $\widehat{s}_j = \alpha_j \oplus \beta_j \oplus \eta_j \oplus \xi_j \oplus \left(A_j^{(l)} \otimes x \otimes \widehat{b}_{j+1} \oplus C_{j+1}^{(l)} \otimes \widehat{b}_{j+2} \oplus a_j^{(l)}\right)$
    $\epsilon\widehat{b}_j = A_j^{(h)} \otimes x \otimes \epsilon\widehat{b}_{j+1} \oplus C_{j+1}^{(h)} \otimes \epsilon\widehat{b}_{j+2} \oplus \widehat{s}_j$
end
$res = y \oplus \epsilon\widehat{b}_0$
```

Algorithm 6: Compensated Clenshaw algorithm.

```
function $b = Convert(a)$
$a, b$ are the vectors of monomial polynomial and orthogonal polynomial's coefficients, respectively
$b^{(n)} = 0,$
$b_0^{(n)} = a_n,$
for $j = n-1 : -1 : 0$
    for $k = 0 : 1 : n-j$
        $b_0^{(j)} = a_j - \frac{C_1}{A_1} b_1^{(j+1)} (k = 0)$
        $b_k^{(j)} = \frac{1}{A_{k-1}} b_{k-1}^{(j+1)} - \frac{C_{k+1}}{A_{k+1}} b_{k+1}^{(j+1)} \quad (k = 1, \ldots, n-j-2)$
        $b_k^{(j)} = \frac{1}{A_{k-1}} b_{k-1}^{(j+1)} \quad (k \geq n-j-1)$
    end
end
$b = b^{(0)}$
```

Algorithm 7: The conversion algorithm from power basis to Legendre basis.

Then, from Definition 1, we get

$$\begin{aligned}\left|\widehat{b}_0 - p(x)\right| &= \left|\left(\prod_{i=0}^{n-1} A_i\right) x^n a_n \theta_{5n-1} + \cdots + a_0\theta_1\right| \\ &\leq \gamma_{5n-1}\left(\left(\prod_{i=0}^{n-1} A_i\right)|x|^n\left|a_n\right| + \cdots + \left|a_0\right|\right) \quad (33) \\ &= \gamma_{5n-1}\sum_{j=0}^{n}\left|a_j\right|\widetilde{P}_j(|x|).\end{aligned}$$

□

Lemma 10. *Let $p(x) = \sum_{j=0}^{n} a_j p_j(x)$ be a Legendre series of degree n and x a floating point value. We assume that $\sigma_j \leq \omega_j(A_j|x||\widehat{b}_{j+1}| + |C_{j+1}||\widehat{b}_{j+2}| + |a_j|)$, where ω_j is real numbers; then*

$$\sum_{j=0}^{n-1}\sigma_j\widetilde{p}_j(|x|) \leq n\omega_j\left(1+\gamma_{5(n-1)}\right)\sum_{j=0}^{n}\left|a_j\right|\widetilde{p}_j(|x|). \quad (34)$$

Proof. According to (31), we get

$$\left|\widehat{b}_j\right| \leq (1+\gamma_5)\left(A_j|x|\left|\widehat{b}_{j+1}\right| + \left|C_{j+1}\right|\left|\widehat{b}_{j+2}\right| + \left|a_j\right|\right). \quad (35)$$

Since $\sigma_j \leq \omega_j(A_j|x||\widehat{b}_{j+1}| + |C_{j+1}||\widehat{b}_{j+2}| + |a_j|)$, from Definition 1, we have

$$\sigma_j \leq \omega_j\left(1+\gamma_{5(n-j-1)}\right)\Sigma_{k=j}^{n}\left|a_k\right|\widetilde{p}_{k-j}(|x|); \quad (36)$$

```
function $[x, y] = FastTwoSum(a, b)$
$x = a \oplus b$
$y = (a \ominus x) \oplus b$
```

Algorithm 8: EFT of the sum of two floating point numbers ($|a| \geq |b|$).

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

Algorithm 9: Addition of a double-double number and a double number.

```
function $[rh, rl] = add\_dd\_dd(ah, al, bh, bl)$
$[sh, sl] = TwoSum(ah, bh)$
$[th, tl] = TwoSum(al, bl)$
$sl = sl \oplus th$
$th = sh \oplus sl$
$sl = sl \ominus (th \ominus sh)$
$tl = tl \oplus sl$
$[rh, rl] = FastTwoSum(th, tl)$
```

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

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

Algorithm 11: Multiplication of a double-double number by a double number.

then

$$\sum_{j=0}^{n-1} \sigma_j \widetilde{p}_j(|x|) \leq \omega_j \left(1 + \gamma_{5(n-1)}\right) \times \sum_{j=0}^{n-1} \left(\sum_{k=j}^{n} |a_k| \widetilde{p}_{k-j}(|x|) \right) \widetilde{p}_j(|x|). \tag{37}$$

According to Lemma 5, we obtain

$$\begin{aligned} \sum_{j=0}^{n-1} \sigma_j \widetilde{p}_j(|x|) &\leq \omega_j \left(1 + \gamma_{5(n-1)}\right) \sum_{j=0}^{n-1} \sum_{k=j}^{n} |a_k| \widetilde{p}_k(|x|) \\ &\leq n\omega_j \left(1 + \gamma_{5(n-1)}\right) \sum_{j=0}^{n} |a_j| \widetilde{p}_j(|x|). \end{aligned} \tag{38}$$

□

Among the perturbed and round-off errors in Algorithm 5, we deem that some errors do not influence the numerical result in working precision. Now we can give the perturbed error bounds and the round-off error bounds, respectively. At first we analyze the perturbed error in (24). Let $g_j^{(l)} = A_j^{(l)} x \widehat{b}_{j+1} + C_{j+1}^{(l)} \widehat{b}_{j+2} + a_j^{(l)}$. According to $A^{(l)} \leq uA^{(h)}$ and Remark 7, we obtain

$$\left| g_j^{(l)} \right| \leq u (1 + \gamma_1) \left(A_j |x| \left| \widehat{b}_{j+1} \right| + \left| C_{j+1} \right| \left| \widehat{b}_{j+2} \right| + \left| a_j \right| \right). \tag{39}$$

Then let $\omega_j = u(1 + \gamma_1)$ and from Lemma 10, taking into account that $(n+1)u(1+\gamma_{5n+1}) \leq (5n+1)u(1+\gamma_{5n+1}) = \gamma_{5n+1}$, we have

$$\left| \sum_{j=0}^{n} g_j^{(l)} p_j(x) \right| \leq \gamma_{5n+1} \sum_{j=0}^{n} \left| a_j \right| \widetilde{p}_j(|x|). \tag{40}$$

γ_{5n+1} is $O(u)$, so this coefficient perturbation may influence the accuracy; we need to consider it in our compensated algorithm.

Similarly, we let $g_j^{(m)} = A_j^{(m)} x \widehat{b}_{j+1} + C_{j+1}^{(m)} \widehat{b}_{j+2} + a_j^{(m)}$; then

$$\left| \sum_{j=0}^{n} g_j^{(m)} p_j(x) \right| \leq u\gamma_{5n+1} \sum_{j=0}^{n} \left| a_j \right| \widetilde{p}_j(|x|). \tag{41}$$

$u\gamma_{5n+1}$ is $O(u^2)$, so this coefficient perturbation does not influence the accuracy.

Remark 11. When $j = n$, the nth step of Algorithm 5 is $\widehat{b}_n = a_n^{(h)}$, so that we only need to consider the perturbation of coefficient a_n.

Next we deduce the round-off error bound. Let $s_j = \alpha_j + \beta_j + \eta_j + \xi_j$. According to Theorems 2 and 3 and Remark 7, we obtain

$$\begin{aligned} \left|\alpha_j\right| &\leq \left(2u + 14u^2\right) \left| A_j^{(h)} x \widehat{b}_{j+1} \right| \\ &\leq \left(2u + 14u^2\right) (1 + \gamma_1) \left| A_j x \widehat{b}_{j+1} \right|; \\ \left|\beta_j\right| &\leq u \left| C_{j+1}^{(h)} \widehat{b}_{j+2} \right| \leq u (1 + \gamma_1) \left| C_{j+1} \widehat{b}_{j+2} \right|; \\ \left|\eta_j\right| &\leq u \left| A_j^{(h)} x \widehat{b}_{j+1} \langle 3 \rangle + C_{j+1}^{(h)} \widehat{b}_{j+2} \langle 2 \rangle \right| \\ &\leq u (1 + \gamma_3) \left| A_j x \widehat{b}_{j+1} + C_{j+1} \widehat{b}_{j+2} \right|; \\ \left|\xi_j\right| &\leq u \left| A_j^{(h)} x \widehat{b}_{j+1} \langle 4 \rangle + C_{j+1}^{(h)} \widehat{b}_{j+2} \langle 3 \rangle + a_j \langle 1 \rangle \right| \\ &\leq u (1 + \gamma_4) \left| A_j x \widehat{b}_{j+1} + C_{j+1} \widehat{b}_{j+2} + a_j \right|; \end{aligned} \tag{42}$$

then

$$\begin{aligned} &\left|\alpha_j\right| + \left|\beta_j\right| + \left|\eta_j\right| + \left|\xi_j\right| \\ &\quad \leq \left(4u + 14u^2\right) \left|1 + \gamma_4\right| \\ &\quad\quad \times \left(A_j |x| \left| \widehat{b}_{j+1} \right| + \left| C_{j+1} \right| \left| \widehat{b}_{j+2} \right| + \left| a_j \right| \right). \end{aligned} \tag{43}$$

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

Algorithm 12: Multiplication of a double-double number by a double-double number.

```
function $[rh, rl] = div\_d\_d(a, b)$
$qh = a \oslash b$
$[th1, tl1] = TwoProd(qh, b)$
$[th2, tl2] = TwoSum(a, -th1)$
$ql = (th2 \oplus tl2 \ominus tl1) \oslash b$
$[rh, rl] = FastTwoSum(qh, ql)$
```

Algorithm 13: Division of a double number by a double number.

Let $\omega_j = (4u + 14u^2)|1+\gamma_4|$ and using it into Lemma 10, from $(4+14u)nu(1+\gamma_{5n-1}) \le (5n-1)u(1+\gamma_{5n-1}) = \gamma_{5n-1}$ $(n \ge 2)$, we have

$$\left|\sum_{j=0}^{n-1}\left(\left|\alpha_j\right| + \left|\beta_j\right| + \left|\eta_j\right| + \left|\xi_j\right|\right) p_j(x)\right| \le \gamma_{5n-1}\sum_{j=0}^{n}\left|a_j\right| \widetilde{p}_j(|x|). \tag{44}$$

γ_{5n-1} is $O(u)$, so this round-off error may influence the accuracy, we also need to consider it in our compensated algorithm.

From Theorem 3 we have known that round-off error $|\tau_j| \le \gamma_2\gamma_6|A_j^{(h)}x\widehat{b}_{j+1}| \le \gamma_2\gamma_6(1+\gamma_1)|A_jx\widehat{b}_{j+1}|$. According to Lemma 10 we let $\omega_j = \gamma_2\gamma_6(1+\gamma_1)$, from $\gamma_2 \le 2u(1+\gamma_1)$ and $2nu(1+\gamma_{5n-3}) \le \gamma_{5n-3}$, we have

$$\left|\sum_{j=0}^{n-1}\right| \tau_j \left|p_j(x)\right| \le \gamma_6\gamma_{5n-3}\sum_{j=0}^{n}\left|a_j\right| \widetilde{p}_j(|x|). \tag{45}$$

$\gamma_6\gamma_{5n-3}$ is $O(u^2)$, so this round-off error does not influence the accuracy.

Observing the error bounds we described above, the perturbation generated by the third part of coefficients in (18) does not influence the accuracy. Thanks to the *div_d_d* algorithm in Appendix B (see Algorithm 13), we only need to split the coefficients into two floating point numbers.

Applying EFT, the *ThreeProd* algorithm and the *div_d_d* algorithm, considering all errors which may influence the numerical result in working precision in Algorithm 5, we obtain the compensated Clenshaw algorithm in Algorithm 6.

Here we give the error bound of Algorithm 6.

Theorem 12. *Let $p(x) = \sum_{j=0}^{n} a_j p_j(x)$ be a Legendre series of degree n $(n \ge 2)$ and x a floating point value. The forward error bound of the compensated Clenshaw algorithm is*

$$\begin{aligned}&\left|CompClenshaw(p,x) - p(x)\right| \\ &\quad \le u\left|p(x)\right| + 2\gamma_{5n-2}^2\sum_{j=0}^{n} a_j P_j(x).\end{aligned} \tag{46}$$

Proof. From Algorithm 6, we obtain

$$\begin{aligned}&\left|CompClenshaw(p,x) - p(x)\right| \\ &\quad = \left|\left(\widehat{b}_0 \oplus \epsilon\widehat{b}_0\right) - p(x)\right| \\ &\quad = \left|(1+\epsilon)\left(\widehat{b}_0 + \epsilon\widehat{b}_0\right) - p(x)\right|,\end{aligned} \tag{47}$$

where $|\epsilon| \le u$.

According to Theorem 8, we have $p(x) = \widehat{b}_o + \epsilon b_o$; then

$$\begin{aligned}&\left|CompClenshaw(p,x) - p(x)\right| \\ &\quad = \left|(1+\epsilon)\left(P(x) - \epsilon b_o + \epsilon\widehat{b}_0\right) - p(x)\right| \\ &\quad \le u\left|P(x)\right| + (1+u)\left|\epsilon b_o - \epsilon\widehat{b}_0\right|.\end{aligned} \tag{48}$$

Next we analyze the bound of $|\epsilon b_o - \epsilon\widehat{b}_0|$.

Let $g_j^{(l)} = A_j^{(l)}x\widehat{b}_{j+1} + C_{j+1}^{(l)}\widehat{b}_{j+2} + a_j^{(l)}$; then

$$\widehat{g}_j^{(l)} = A_j^{(l)}x\widehat{b}_{j+1}\langle 4\rangle + C_{j+1}^{(l)}\widehat{b}_{j+2}\langle 3\rangle + a_j^{(l)}\langle 1\rangle. \tag{49}$$

Thus we obtain

$$\begin{aligned}&\left|\sum_{j=0}^{n} g_j^{(l)} p_j(x) - \sum_{j=0}^{n}\widehat{g}_j^{(l)} p_j(x)\right| \\ &\quad \le \gamma_4\sum_{j=0}^{n}\left(A_j^{(l)}|x|\left|\widehat{b}_{j+1}\right| + \left|C_{j+1}^{(l)}\right|\left|\widehat{b}_{j+2}\right| + \left|a_j^{(l)}\right|\right)\widetilde{p}_j(|x|).\end{aligned} \tag{50}$$

According to Lemma 9, we get

$$\begin{aligned}&\left|\sum_{j=0}^{n}\widehat{g}_j^{(l)} p_j(x) - \oplus\sum_{j=0}^{n}\widehat{g}_j^{(l)} \otimes p_j(x)\right| \\ &\quad \le \gamma_{5n-1}\sum_{j=0}^{n}\left|\widehat{g}_j^{(l)}\right|\widetilde{p}_j(|x|) \\ &\quad \le \gamma_{5n-1}(1+\gamma_4) \\ &\qquad \times\sum_{j=0}^{n}\left(A_j^{(l)}|x|\left|\widehat{b}_{j+1}\right| + \left|C_{j+1}^{(l)}\right|\left|\widehat{b}_{j+2}\right| + \left|a_j^{(l)}\right|\right)\widetilde{p}_j(|x|).\end{aligned} \tag{51}$$

```
function $[rh, rl] = div\_dd\_dd(ah, al, bh, bl)$
$q1 = ah \oslash bh$
$[th1, tl1] = prod\_dd\_d(bh, bl, q1)$
$[th2, tl2] = add\_dd\_dd(ah, al, -th1, -tl1)$
$q2 = th2 \oslash bh$
$[th1, tl1] = prod\_dd\_d(bh, bl, q2)$
$[th2, tl2] = add\_dd\_dd(th2, tl2, -th1, -tl1)$
$q3 = th2 \oslash bh$
$[q1, q2] = FastTwoSum(q1, q2)$
$[rh, rl] = add\_dd\_d(q1, q2, q3)$
```

Algorithm 14: Division of a double-double number by a double-double number.

```
function $res = DDClenshaw(p, x)$
$b_{n+2} = b_{n+1} = 0$
for $j = n : -1 : 0$
    $[h1, l1] = prod\_dd\_d(b_{j+1}^{(h)}, b_{j+1}^{(l)}, x)$
    $[h2, l2] = prod\_dd\_dd(h1, l1, A_j^{(h)}, A_j^{(l)})$
    $[h3, l3] = prod\_dd\_dd(b_{j+2}^{(h)}, b_{j+2}^{(l)}, C_{j+1}^{(h)}, C_{j+1}^{(l)})$
    $[h4, l4] = add\_dd\_dd(h2, l2, h3, l3)$
    $[b_j^{(h)}, b_j^{(l)}] = add\_dd\_dd(h4, l4, a_j^{(h)}, a_j^{(l)})$
end
$res = [b_0^{(h)}, b_0^{(l)}]$
```

Algorithm 15: *DDClenshaw* algorithm of evaluating an Legendre series in double-double arithmetic.

From (50), (51) and $\gamma_{5n-1}(1+\gamma_4)+\gamma_4 \le \gamma_{5n+3}$, we derive

$$\begin{aligned}\left|\sum_{j=0}^{n} g_j^{(l)} p_j(x) - \oplus\sum_{j=0}^{n}\widehat{g}_j^{(l)} \otimes p_j(x)\right| &\le \left|\sum_{j=0}^{n} g_j^{(l)} p_j(x) - \sum_{j=0}^{n}\widehat{g}_j^{(l)} p_j(x)\right| \\ &\quad + \left|\sum_{j=0}^{n}\widehat{g}_j^{(l)} p_j(x) - \oplus\sum_{j=0}^{n}\widehat{g}_j^{(l)} \otimes p_j(x)\right| \\ &\le \gamma_{5n+3}\sum_{j=0}^{n}\left(A_j^{(l)}|x|\left|\widehat{b}_{j+1}\right| + \left|C_{j+1}^{(l)}\right|\left|\widehat{b}_{j+2}\right| + \left|a_j^{(l)}\right|\right)\widetilde{p}_j(|x|).\end{aligned} \tag{52}$$

According to $A^{(l)} \le uA^{(h)}$ and Remark 7, we obtain

$$\begin{aligned}&A_j^{(l)}|x|\left|\widehat{b}_{j+1}\right| + \left|C_{j+1}^{(l)}\right|\left|\widehat{b}_{j+2}\right| + \left|a_j^{(l)}\right| \\ &\quad \le u(1+\gamma_1)\left(A_j|x|\left|\widehat{b}_{j+1}\right| + \left|C_{j+1}\right|\left|\widehat{b}_{j+2}\right| + \left|a_j\right|\right).\end{aligned} \tag{53}$$

From (53) we use $\omega_j = u(1+\gamma_1)$ into Lemma 10; taking into account that $nu(1+\gamma_{5n-4}) \le (5n+1)u(1+\gamma_{5n+1}) = \gamma_{5n+1}$ and $\gamma_{5n+1}\gamma_{5n+3} \le \gamma_{5n+2}^2$, according to (52), we get

$$\begin{aligned}&\left|\sum_{j=0}^{n-1} g_j^{(l)} p_j(x) - \oplus\sum_{j=0}^{n-1}\widehat{g}_j^{(l)} \otimes p_j(x)\right| \\ &\quad \le \gamma_{5n+2}^2\sum_{j=0}^{n}\left|a_j\right|\widetilde{p}_j(|x|).\end{aligned} \tag{54}$$

Next we let $s_j = \alpha_j + \beta_j + \eta_j + \xi_j$; then

$$\widehat{s}_j \le (1+\gamma_3)\left(\left|\alpha_j\right| + \left|\beta_j\right| + \left|\eta_j\right| + \left|\xi_j\right|\right); \tag{55}$$

thus

$$\begin{aligned}&\left|\sum_{j=0}^{n-1} s_j p_j(x) - \sum_{j=0}^{n-1}\widehat{s}_j p_j(x)\right| \\ &\quad \le \gamma_3\sum_{j=0}^{n-1}\left(\left|\alpha_j\right| + \left|\beta_j\right| + \left|\eta_j\right| + \left|\xi_j\right|\right)\widetilde{p}_j(|x|).\end{aligned} \tag{56}$$

According to Lemma 9, we obtain

$$\begin{aligned}&\left|\sum_{j=0}^{n-1}\widehat{s}_j p_j(x) - \oplus\sum_{j=0}^{n-1}\widehat{s}_j \otimes p_j(x)\right| \\ &\quad \le \gamma_{5(n-1)-1}\sum_{j=0}^{n-1}\left|\widehat{s}_j\right|\widetilde{p}_j(|x|) \\ &\quad \le \gamma_{5(n-1)-1}(1+\gamma_3) \\ &\qquad \times \sum_{j=0}^{n-1}\left(\left|\alpha_j\right| + \left|\beta_j\right| + \left|\eta_j\right| + \left|\xi_j\right|\right)\widetilde{p}_j(|x|).\end{aligned} \tag{57}$$

From (56), (57), and $\gamma_{5(n-1)-1}(1+\gamma_3)+\gamma_3 \le \gamma_{5n-3}$, we derive

$$\begin{aligned}\left|\sum_{j=0}^{n-1} s_j p_j(x) - \oplus\sum_{j=0}^{n-1}\widehat{s}_j \otimes p_j(x)\right| &\le \left|\sum_{j=0}^{n-1} s_j p_j(x) - \sum_{j=0}^{n-1}\widehat{s}_j p_j(x)\right| \\ &\quad + \left|\sum_{j=0}^{n-1}\widehat{s}_j p_j(x) - \oplus\sum_{j=0}^{n-1}\widehat{s}_j \otimes p_j(x)\right| \\ &\le \gamma_{5n-3}\sum_{j=0}^{n-1}\left(\left|\alpha_j\right| + \left|\beta_j\right| + \left|\eta_j\right| + \left|\xi_j\right|\right)\widetilde{p}_j(|x|).\end{aligned} \tag{58}$$

From (43), we let $\omega_j = (4+14u)u|1+\gamma_4|$ and using it into Lemma 10, taking into account that $(4+14u)nu(1+\gamma_{5n-1}) \le (5n-1)u(1+\gamma_{5n-1}) = \gamma_{5n-1}(n \ge 2)$ and $\gamma_{5n-3}\gamma_{5n-1} \le \gamma_{5n-2}^2$, we have

$$\left|\sum_{j=0}^{n-1} s_j p_j(x) - \oplus\sum_{j=0}^{n-1}\widehat{s}_j \otimes p_j(x)\right| \le \gamma_{5n-2}^2\sum_{j=0}^{n-1}\left|a_j\right|\widetilde{p}_j(|x|). \tag{59}$$

```
function $b = DDConvert(a)$
$a, b$ are the vectors of monomial polynomial and orthogonal polynomial's coefficients, respectively
$b^{(n)} = 0$,
$bh_0^{(n)} = a_n^{(h)}$,
for $j = n-1 : -1 : 0$
    for $k = 0 : 1 : n-j$
        if $k = 0$
      $[h_1, l_1] = prod\_dd\_dd(C_1^{(h)} C_1^{(l)}, bh_1^{(j+1)}, bl_1^{(j+1)})$
      $[h_2, l_2] = div\_dd\_dd(h_1, l_1, A_1^{(h)}, A_1^{(l)})$
      $[bh_0^{(j)}, bl_0^{(j)}] = add\_dd\_dd(-h_2, -l_2, a_j^{(h)}, a_j^{(l)})$
        if $k = 1, 2, \ldots, n-j-2$
      $[h_1, l_1] = div\_dd\_dd(bh_{k-1}^{(j+1)}, bl_{k-1}^{(j+1)}, A_{k-1}^{(h)}, A_{k-1}^{(l)})$
      $[h_2, l_2] = prod\_dd\_dd(C_{k+1}^{(h)}, C_{k+1}^{(l)}, bh_{k+1}^{(j+1)}, bl_{k+1}^{(j+1)})$
      $[h_3, l_3] = div\_dd\_dd(h_2, l_2, A_{k+1}^{(h)}, A_{k+1}^{(l)})$
      $[bh_k^{(j)}, bl_k^{(j)}] = add\_dd\_dd(h_1, l_1, -h_3, -l_3))$
        if $k \geq n-j-1$
      $[bh_k^{(j)}, bl_k^{(j)}] = div\_dd\_dd(bh_{k-1}^{(j+1)}, bl_{k-1}^{(j+1)}, A_{k-1}^{(h)}, A_{k-1}^{(l)})$
    end
end
$b = [bh^{(0)}, bl^{(0)}]$
```

Algorithm 16: *DDConvert* algorithm for the polynomial from power basis to Legendre basis.

Combining (54) and (59), we get

$$\begin{aligned}
\left|\epsilon b_o - \epsilon \hat{b}_0\right| &= \left|\sum_{j=0}^{n-1} s_j p_j(x) + \sum_{j=0}^{n-1} g_j^{(l)} p_j(x) \right. \\
&\quad \left. - \oplus \sum_{j=0}^{n-1} \hat{s}_j \otimes p_j(x) - \oplus \sum_{j=0}^{n-1} \hat{g}_j^{(l)} \otimes p_j(x)\right| \\
&\leq \left|\sum_{j=0}^{n-1} s_j p_j(x) - \oplus \sum_{j=0}^{n-1} \hat{s}_j \otimes p_j(x)\right| \\
&\quad + \left|\sum_{j=0}^{n-1} g_j^{(l)} p_j(x) - \oplus \sum_{j=0}^{n-1} \hat{g}_j^{(l)} \otimes p_j(x)\right| \\
&\leq 2\gamma_{5n+2}^2 \sum_{j=0}^{n-1} \left|a_j\right| \tilde{p}_j(|x|).
\end{aligned} \tag{60}$$

□

From Definitions 4 and 6 and Theorem 12, we easily get the following corollary.

Corollary 13. *Let $p(x) = \sum_{j=0}^{n} a_j p_j(x)$ be a Legendre series of degree n and x a floating point value. The relative error bound of the compensated Clenshaw algorithm is*

$$\frac{\left|CompClenshaw(p, x) - p(x)\right|}{\left|p(x)\right|} \leq u + 2\gamma_{5n+2}^2 \operatorname{cond}_{\text{rel}}(p, x). \tag{61}$$

4. Numerical Results

All our experiments are performed using IEEE-754 double precision as working precision. Here, we consider the polynomials in Legendre basis with real coefficients and floating point entry x. All the programs about accuracy measurements have been written in MATLAB R2012b and that about timing measurements have been written in C code on a 2.53-GHz Intel Core i5 laptop.

4.1. Evaluation of the Polynomial in Legendre Basis. In order to construct an ill-conditioned polynomial, we consider the evaluation of the polynomial in Legendre basis $p(x) = \sum_{i=0}^{17} a_i p_i(x)$ converted by the polynomials $P(x) = (x-0.75)^7 (x-1)^{11}$ in the neighborhood of its multiple roots 0.75 and 1. We use the *Clenshaw*, *CompClenshaw* algorithms and the Symbolic Toolbox to evaluate the polynomial in Legendre basis. In order to observe the perturbation of polynomial coefficients clearly, we propose the *Convert* algorithm in Appendix A (see Algorithm 7) to obtain the coefficients of the Legendre series. To decrease the perturbation of the coefficients we also propose the *DDConvert* algorithm in Appendix B (see Algorithm 16) to store the coefficients in double-double format. That is, the coefficients of the polynomial evaluated, which are obtained by the *Convert* algorithm and *DDConvert* algorithm, are in double format and double-double format, respectively. In this experiment we evaluate the polynomials for 400 equally spaced points in the intervals $[0.68, 1.15]$, $[0.7485, 0.7515]$, and $[0.993, 1.007]$.

From Figures 1-2 we observe that the polynomial evaluated by *Clenshaw* algorithm (on the top figure) is oscillating, and the compensated algorithm is more smooth drawing. The polynomials we evaluated by Symbolic Toolbox (on the bottom of Figures 1-2) are different because the perturbations of coefficients obtained by the *DDConvert* algorithm are smaller than those by the *Convert* algorithm. We can see that the accuracy of the polynomials evaluated by the compensated algorithm is the same with evaluated by Symbolic

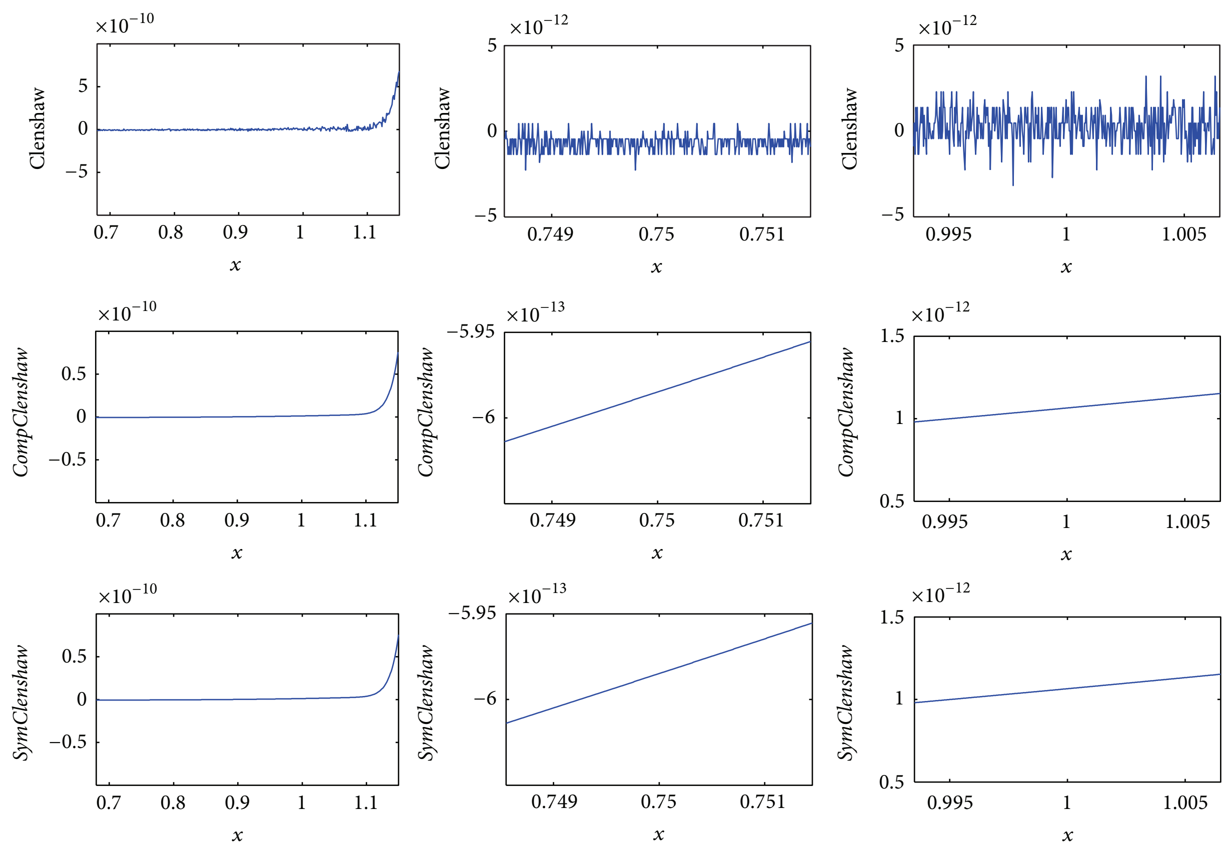

FIGURE 1: Evaluation of polynomial converted from $p(x) = (x-0.75)^7(x-1)^{11}$ by the *Convert* algorithm in Legendre basis in the neighborhood of its multiple roots, using the Clenshaw algorithm (up), the *CompClenshaw* algorithm (middle), and Symbolic Toolbox (down).

Toolbox in Figure 1. The polynomials in Legendre basis evaluated by the compensated algorithm are much more smooth drawing and just a little oscillation in the intervals [0.7485, 0.7515] in Figure 2. In fact, if we use the Symbolic Toolbox to get the polynomial coefficients, the oscillation will be smaller than it is in Figure 2. However, this method is expensive. We just need to use the *DDConvert* algorithm to get the coefficients; the result obtained by the *CompClenshaw* algorithm is almost the same as that by using the Symbolic Toolbox in working precision.

4.2. Accuracy of the Compensated Algorithm. The closer to the root, the larger the condition number. Thus, in this experiment, the evaluation is for 120 points near the root 0.75, that is, $x = 0.75 - 1.03^{2i-85}$, for $i = 1:40$ and $x = 0.75 - 1.13^{i-85}$ for $i = 1:80$. We compare the compensated algorithm with multiple precision library. Since the working precision is double precision, we choose the double-double arithmetic [22] (see Appendix B) which is the most efficient way to yield a full precision accuracy of evaluating the polynomial in Legendre basis to compare with the compensated algorithm. We evaluate the polynomials by the *Clenshaw*, *CompClenshaw*, and *DDClenshaw* algorithms in Appendix B (see Algorithm 15) and the Symbolic Toolbox, respectively, so that the relative forward errors can be obtained by $|p_{\text{res}}(x) - p_{\text{sym}}(x)|/p_{\text{sym}}(x)$ and the relative error bounds are described from Corollaries 13 and A.2 in Appendix A. Then we propose the relative forward errors of evaluation of the polynomial in Legendre basis in Figure 3. As we can see, the relative errors of the compensated algorithm and double-double arithmetic are both smaller than u ($u \approx 1.16 \times 10^{-16}$) when the condition number is less than 10^{17}. And the accuracy of both algorithms is decreasing linearly for the condition number larger than 10^{17}. However, the *Clenshaw* algorithm cannot yield the working precision; the accuracy of which decreases linearly since the condition number is less than 10^{17}. When the condition number is lager than 10^{17}, the Clenshaw algorithm cannot obtain even one significant bit.

4.3. Time Performances. We can easily know that the *TwoSum*, *TwoProd*, and *FastTwoSum* algorithms in Appendix B (Algorithm 8) require 6, 17, and 3 flops, respectively. Then we obtain the computational cost of the *Clenshaw*, *CompClenshaw*, and *DDClenshaw* algorithms:

(i) *Clenshaw*: 5n-2 flops;

(ii) *CompClenshaw*: 79n-29 flops;

(iii) *DDClenshaw*: 110n-44 flops.

Considering the previous comparison of the accuracy, we observe that *CompClenshaw* is as accurate as *DDClenshaw* in double precision, but it only needs about 71.8% of flops

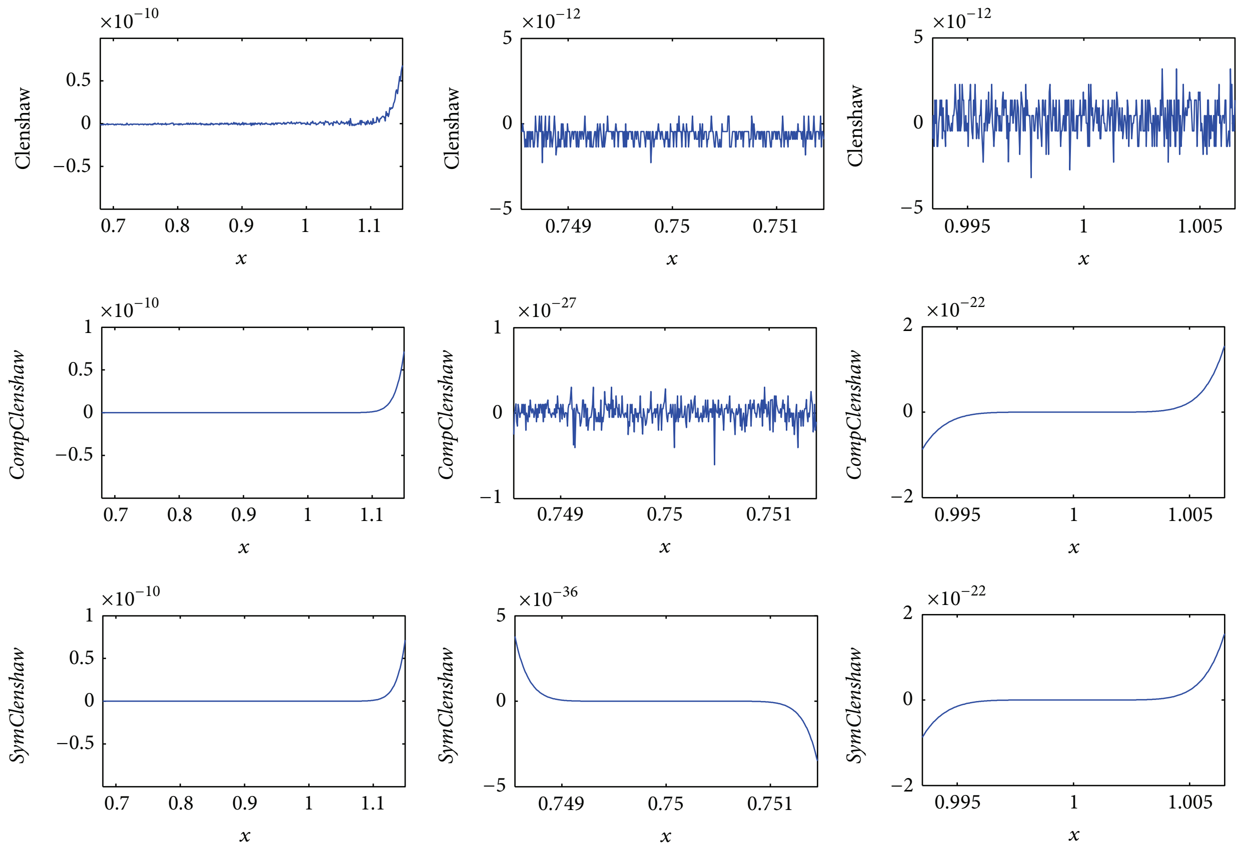

FIGURE 2: Evaluation of polynomial converted from $p(x) = (x - 0.75)^7(x - 1)^{11}$ by the *DDConvert* algorithm in Legendre basis in the neighborhood of its multiple roots, using the Clenshaw algorithm (up), the *CompClenshaw* algorithm (middle), and Symbolic Toolbox (down).

counting on average. We also implement *CompClenshaw* and *DDClenshaw* by using Microsoft Visual C++ 2008 on Windows 7. Similar to the statement in [23], we assume that the computing time of these algorithms does not depend on the coefficients of polynomial in Legendre basis nor the argument x. So we generate the tested polynomials with random coefficients and arguments in the interval $(-1, 1)$, whose degrees vary from 20 to 10000 by the step 50. The average measured computing time ratio of *CompClenshaw* to *DDClenshaw* in C code is 58.29%. The reason why the measured computing time ratio is better than the theoretical flop count one can be referred to the analysis in terms of instruction level parallelism (ILP) described in [24, 25].

5. Conclusions

This paper introduces a compensated Clenshaw algorithm for accurate evaluation of the finite Legendre series. The *Clenshaw* algorithm is not precise enough for an ill-conditioned problem, especially evaluating a polynomial in the neighborhood of a multiple root. However, this new algorithm can yield a full precision accuracy in working precision, as the same as the original Clenshaw algorithm using double-double arithmetic and rounding into the working precision. Meanwhile this compensated Clenshaw algorithm is more efficient which means that it is much more useful to accurately evaluate the polynomials in Legendre basis for ill-conditioned situations.

Appendices

A. The Coefficients Conversion Algorithm from Polynomial in Power Basis to Polynomial in Legendre Basis

In order to design ill-conditioned problem of the polynomial evaluation, we evaluate a polynomial in the neighborhood of a multiple root. Thus we need an algorithm for converting a polynomial in power basis to the polynomial in Legendre basis. Motivated by [26], the conversion algorithm can be deduced as follows.

Let $P(x) = \sum_{k=0}^{n} a_k x^k$ and $p(x) = \sum_{k=0}^{n} b_k^{(0)} p_k$ be the polynomials in power basis and Legendre basis, respectively; according to

$$\begin{gathered} p_{n+1} = A_n x p_n + C_n p_{n-1} \\ \Longrightarrow x p_n = \frac{1}{A_n} p_{n+1} - \frac{C_n}{A_n} p_{n-1} \quad (n > 1), \\ p_1 = A_0 x p_0 \Longrightarrow x p_0 = \frac{1}{A_0} p_1, \end{gathered} \tag{A.1}$$

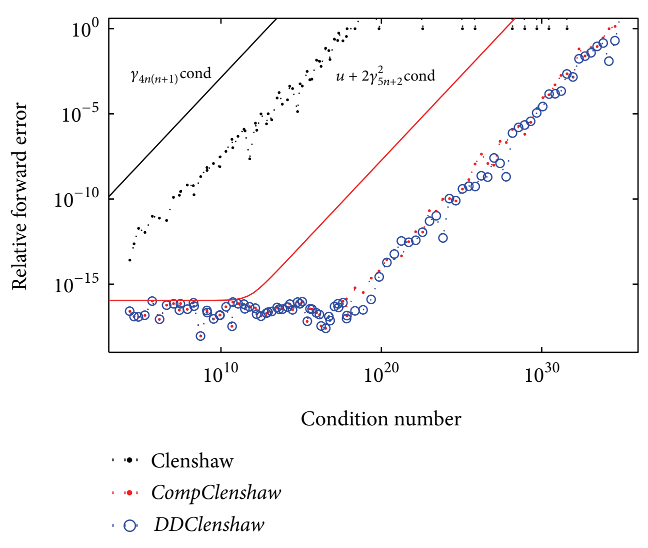

FIGURE 3: Accuracy of evaluation of polynomial converted from $p(x) = (x-0.75)^7(x-1)^{11}$ by the *DDConvert* algorithm in Legendre basis with respect to the condition number.

we have

$$p_n = \sum_{k=0}^{j-1} a_k x^k + x^j \sum_{k=0}^{n-j} b_k^{(j)} p_k(x) \equiv \sum_{k=0}^{j} a_k x^k$$

$$+ x^j \sum_{k=1}^{n-j-1} b_k^{(j+1)} \left(\frac{1}{A_k} p_{k+1} - \frac{C_k}{A_k} p_{k-1} \right) + x^{j+1} b_0^{(j+1)} p_0. \quad \text{(A.2)}$$

Thus we get Algorithm 7.

Theorem A.1. *Let $P(x) = \sum_{k=0}^{n} a_k x^k$ and $p(x) = \sum_{k=0}^{n} b_k^{(0)} p_k$ be the polynomials in power basis and Legendre basis, respectively. The forward error bound of Algorithm 7 is*

$$\left|\hat{b}^{(0)} - b^{(0)}\right| \le \gamma_{4n-1} \left|b^{(0)}\right|. \quad \text{(A.3)}$$

Proof. According to Definition 1 and Remark 7, we have

$$\begin{aligned} b_0^{(j)} &= a_j \langle 1 \rangle - \frac{C_1}{A_1} b_1^{(j+1)} \langle 5 \rangle \quad (k = 0), \\ b_k^{(j)} &= \frac{1}{A_{k-1}} b_{k-1}^{(j+1)} \langle 3 \rangle - \frac{C_{k+1}}{A_{k+1}} b_{k+1}^{(j+1)} \langle 5 \rangle \\ &\qquad (k = 1, \dots, n-j-2), \\ b_k^{(j)} &= \frac{1}{A_{k-1}} b_{k-1}^{(j+1)} \langle 2 \rangle \quad (k \ge n-j-1). \end{aligned} \quad \text{(A.4)}$$

When $j = n-1$, we get that

$$\begin{aligned} b_0^{(n-1)} &= a_{n-1}, \\ b_1^{(n-1)} &= \frac{a_n}{A_0} \langle 2 \rangle. \end{aligned} \quad \text{(A.5)}$$

Hence we can obtain

$$\begin{array}{c|cccccccc} j \setminus k & 0 & 1 & 2 & 3 & \cdots & n-2 & n-1 \\ \hline n-1 & & \langle 2 \rangle & 0 & 0 & \cdots & 0 & 0 \\ n-2 & \langle 7 \rangle & \langle 2 \rangle & \langle 4 \rangle & 0 & \cdots & 0 & 0 \\ n-3 & \langle 7 \rangle & \langle 10 \rangle & \langle 4 \rangle & \langle 6 \rangle & \cdots & 0 & 0 \\ n-4 & \langle 15 \rangle & \langle 10 \rangle & \langle 13 \rangle & \langle 6 \rangle & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & \left\langle 8\left[\frac{n}{2}\right] - 1 \right\rangle & \left\langle 8\left[\frac{n-1}{2}\right] + 2 \right\rangle & \left\langle 8\left[\frac{n}{2}\right] - 3 \right\rangle & \left\langle 8\left[\frac{n-1}{2}\right] \right\rangle & \cdots & \langle 2n-2 \rangle & \langle 2n \rangle \end{array}. \quad \text{(A.6)}$$

Then we have

$$\left|\hat{b}^{(0)} - b^{(0)}\right| \le \gamma_{\max\{8[n/2]-1, 8[(n-1)/2]+2\}} \left|b^{(0)}\right|. \quad \text{(A.7)}$$

According to the properties of Definition 1, we obtain

$$\gamma_{\max\{8[n/2]-1, 8[(n-1)/2]+2\}} \le \gamma_{4n-1}; \quad \text{(A.8)}$$

thus

$$\left|\hat{b}^{(0)} - b^{(0)}\right| \le \gamma_{4n-1}. \quad \text{(A.9)}$$

□

According to Theorem A.1 and Theorem 3.5 in [21], we obtain the following corollary.

Corollary A.2. *Let* $P(x) = \sum_{k=0}^{n} a_k x^k$ *and* $p(x) = \sum_{k=0}^{n} b_k^{(0)} p_k$ *be the polynomials in power basis and Legendre basis, respectively. Then the relative error bound of Clenshaw algorithm by using the Covert algorithm is*

$$\frac{|Clenshaw\,(p,x) - p\,(x)|}{|p\,(x)|} \le \gamma_{4n(n+1)} \mathrm{cond}_{\mathrm{rel}}\,(p,x)\,. \quad (A.10)$$

B. Double-Double Library

The double-double arithmetic is based on Algorithms 8–14 [27]. Algorithms 15 and 16 are Algorithms 5 and 7 in double-double arithmetic, respectively.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by the Science Project of National University of Defense Technology (JC120201) and the National Natural Science Foundation of Hunan Province in China (13JJ2001).

References

[1] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, Chapman & Hall/CRC, Boca Raton, Fla, USA, 2003.

[2] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, vol. 26 of *CBMS-NSF Regional Conference Series in Applied Mathematics*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 1977.

[3] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, Pennsylvania, 2013.

[4] C. W. Clenshaw, "A note on the summation of Chebyshev series," *Mathematical Tables and Other Aids to Computation*, vol. 9, pp. 118–120, 1955.

[5] R. Barrio, "A unified rounding error bound for polynomial evaluation," *Advances in Computational Mathematics*, vol. 19, no. 4, pp. 385–399, 2003.

[6] J. Oliver, "Rounding error propagation in polynomial evaluation schemes," *Journal of Computational and Applied Mathematics*, vol. 5, no. 2, pp. 85–97, 1979.

[7] R. Barrio, "A matrix analysis of the stability of the Clenshaw algorithm," *Extracta Mathematicae*, vol. 13, no. 1, pp. 21–26, 1998.

[8] R. Barrio, "Rounding error bounds for the Clenshaw and Forsythe algorithms for the evaluation of orthogonal polynomial series," *Journal of Computational and Applied Mathematics*, vol. 138, no. 2, pp. 185–204, 2002.

[9] M. Skrzipek, "Polynomial evaluation and associated polynomials," *Numerische Mathematik*, vol. 79, no. 4, pp. 601–613, 1998.

[10] A. Smoktunowicz, "Backward stability of Clenshaw's algorithm," *BIT: Numerical Mathematics*, vol. 42, no. 3, pp. 600–610, 2002.

[11] T. Ogita, S. M. Rump, and S. Oishi, "Accurate sum and dot product," *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 1955–1988, 2005.

[12] S. Graillat, P. Langlois, and N. Louvet, "Compensated Horner scheme," Research Report RR2005-04, LP2A, University of Perpignan, Perpignan, France, 2005.

[13] H. Jiang, S. Graillat, C. Hu et al., "Accurate evaluation of the k-th derivative of a polynomial and its application," *Journal of Computational and Applied Mathematics*, vol. 243, pp. 28–47, 2013.

[14] H. Jiang, S. Li, L. Cheng, and F. Su, "Accurate evaluation of a polynomial and its derivative in Bernstein form," *Computers & Mathematics with Applications*, vol. 60, no. 3, pp. 744–755, 2010.

[15] 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*, vol. 217, no. 23, pp. 9702–9716, 2011.

[16] D. H. Bailey, R. Barrio, and J. M. Borwein, "High-precision computation: mathematical physics and dynamics," *Applied Mathematics and Computation*, vol. 218, no. 20, pp. 10106–10121, 2012.

[17] N. J. Higham, *Accuracy and Stability of Numerical Algorithm*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 2nd edition, 2002.

[18] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, Addison-Wesley, Reading, Mass, USA, 3rd edition, 1998.

[19] T. J. Dekker, "A floating-point technique for extending the available precision," *Numerische Mathematik*, vol. 18, pp. 224–242, 1971.

[20] S. Graillat, "Accurate floating-point product and exponentiation," *IEEE Transactions on Computers*, vol. 58, no. 7, pp. 994–1000, 2009.

[21] R. Barrio, H. Jiang, and S. Serrano, "A general condition number for polynomials," *SIAM Journal on Numerical Analysis*, vol. 51, no. 2, pp. 1280–1294, 2013.

[22] X. Li, J. W. Demmel, D. H. Bailey et al., "Design, implementation and testing of extended and mixed precision BLAS," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 152–205, 2002.

[23] S. Graillat, P. Langlois, and N. Louvet, "Algorithms for accurate, validated and fast polynomial evaluation," *Japan Journal of Industrial and Applied Mathematics*, vol. 26, no. 2-3, pp. 191–214, 2009.

[24] N. Louvet, *Compensated algorithms in floating point arithmetic: accuracy, validation, performances [Ph.D. thesis]*, Universite' de Perpignan Via Domitia, 2007.

[25] P. Langlois and N. Louvet, "More instruction level parallelism explains the actual efficiency of compensated algorithms," Tech. Rep. hal-00165020, DALI Research Team, University of Perpignan, Perpignan, France, 2007.

[26] "Table of Contents for MATH77/mathc90," chapter 11.3 from, http://mathalacarte.com/c/math77_head.html.

[27] Y. Hida, X. S. Li, and D. H. Bailey, "Algorithms for quad-double precision floating point arithmetic," in *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pp. 155–162, Vail, Colo, USA, June 2001.