# Efficient and accurate evaluation of Bézier tensor product surfaces*

Jing Lan[1], Hao Jiang[2**], and Peibing Du[3]

[1] Rongzhi College, Chongqing Technology and Business University, Chongqing, China
[2] College of Computer, National University of Defense Technology, Changsha, China
[3] Northwest Institute of Nuclear Technology, Xi'an, China
haojiang@nudt.edu.cn

**Abstract.** This article proposes a bivariate compensated Volk and Schumaker (`CompVSTP`) algorithm, which extends the compensated Volk and Schumaker (`CompVS`) algorithm, to evaluate Bèzier tensor product surfaces with floating-point coefficients and coordinates. The `CompVSTP` algorithm is obtained by applying error-free transformations to improve the traditional Volk and Schumaker tensor product (`VSTP`) algorithm. We study in detail the forward error analysis of the `VSTP`, `CompVS` and `CompVSTP` algorithms. Our numerical experiments illustrate that the `CompVSTP` algorithm is much more accurate than the `VSTP` algorithm, relegating the influence of the condition numbers up to second order in the rounding unit of the computer.

**Keywords:** Bézier tensor product surfaces · Volk and Schumaker algorithm · Compensated algorithm · Error-free transformation · Round-off error

## 1 Introduction

Tensor product surfaces are bivariate polynomials in tensor product form. In monomial basis, tensor product polynomials are expressed in the following form,

$$p(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} c_{i,j}x^i y^j.$$

In Computer Aided Geometric Design (CAGD), tensor product surfaces are usually represented in Bézier form [1]

$$p(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} c_{i,j}B_i^n(x)B_i^m(y), \quad (x,y) \in [0,1]\times[0,1],$$

* Partially supported by National Natural Science Foundation of China (No. 61402495, No. 61602166), and Chongqing education science planning project 2015-GX-036, which research on the construction for Chongqing smart education.

** Corresponding author

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

where $B_i^k(t)$ is the Bernstein polynomial of degree $k$ as

$$B_i^k(t) = \binom{k}{i}(1-t)^{k-i}t^i, \quad t \in [0,1], \quad i = 0, 1, \ldots, k.$$

The de Casteljau algorithm [2, 3] is the usual polynomial evaluation algorithm in CAGD. Nevertheless, evaluating a polynomial of degree $n$, the de Casteljau algorithm needs $\mathcal{O}(n^2)$ operations, in contrast to the $\mathcal{O}(n)$ operations of the Volk and Schumaker (VS) algorithm [4]. The VS basis $z_n := (z_0^n(t), z_1^n(t), \ldots, z_n^n(t))(t \in [0,1])$ is given by $z_i^n(t) = t^i(1-t)^{n-i}$. Otherwise, the VS algorithm consist of Horner algorithm. For evaluating tensor product surfaces, de Casteljau and VS algorithms are more stable and accurate than Horner algorithm [1]. And these three algorithms satisfy the relative accuracy bound

$$\frac{|p(x,y) - \widehat{p}(x,y)|}{|p(x,y)|} \leq \mathcal{O}(u) \times cond(p,x,y),$$

where $\widehat{p}(x,y)$ is the computed result, $u$ is the unit roundoff and $cond(p,x,y)$ is the condition number of $p(x,y)$.

From 2005 to 2009, Graillat et al proposed compensated Horner scheme for univariate polynomials in [5–7]. From 2010 to 2013, Jiang et al presented compensated de Casteljau algorithms to evaluate univariate polynomials and its first order derivative in Bernstein basis in [8], to evaluate bivariate polynomials in Bernstein-Bézier form in [9], and to evaluate Bézier tensor product surfaces in [10]. From 2014 to 2017, Du et al improved Clenshaw-Smith algorithm [11] for Legendre polynomial series with real number coefficients, bivariate compensated Horner algorithm [12] for tensor product polynomials and the quotient-difference algorithm [13] which is a double nested algorithm. All these algorithms can yield a full precision accuracy in double precision as applying double-double library [14].

This paper presents new compensated VS algorithms, which have less computational cost than compensated de Casteljau algorithm, to evaluate tensor product polynomial surfaces by applying error-free transformations which is exhaustively studied in [15–17]. The relative accuracy bound of our proposed compensated algorithms is satisfied

$$\frac{|p(x,y) - \widehat{p}(x,y)|}{|p(x,y)|} \leq u + \mathcal{O}(u^2) \times cond(p,x,y),$$

where $\widehat{p}(x,y)$ is computed by the compensated algorithms.

The rest of the paper is organized as follows. Section 2 introduces basic notation in error analysis, error-free transformations and condition numbers are also given. Section 3 presents the new compensated VS tensor product algorithm and its error analysis. Finally all the error bounds are compared in numerical experiments in section 4.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

## 2 Preliminary

### 2.1 Basic notations

We assume to work with a floating-point arithmetic adhering to IEEE-754 floating-point standard rounding to nearest. In our analysis we assume that there is no computational overflow or underflow. Let $op \in \{\oplus, \ominus, \otimes, \oslash\}$ represents a floating-point computation, and the evaluation of an expression in floating-point arithmetic is denoted $fl(\cdot)$, then its computation obeys the model

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

where $a, b \in \mathbb{F}$ (the set of floating-point numbers), $\circ \in \{+, -, \times, \div\}$ and $|\varepsilon_1|, |\varepsilon_2| \leq u$ ($u$ is the round-off unit of the computer). We also assume that if $a \circ b = x$ for $x \in \mathbb{R}$, then the computed result in floating-point arithmetic is denoted by $\widehat{x} = a \ op \ b$, and its perturbation is $\epsilon x$, i.e.

$$\widehat{x} = x + \epsilon x. \tag{2}$$

The following definition and properties will be used in the forward error analysis (see more details in [18]).

**Definition 1.** *We define*

$$1 + \theta_n = \prod_{i=1}^{n} (1 + \delta_i)^{\rho_i}, \tag{3}$$

*where* $|\delta_i| \leq u, \rho_i = \pm 1$ *for* $i = 1, 2, \ldots, n$, $|\theta_n| \leq \gamma_n := \dfrac{nu}{1 - nu} = nu + \mathcal{O}(u^2)$ *and* $nu < 1$.

Some basic properties in Definition 1 are given by:

- $u + \gamma_k \leq \gamma_{k+1}$,
- $i\gamma_k < \gamma_{ik}$,
- $\gamma_k + \gamma_j + \gamma_k\gamma_j \leq \gamma_{k+j}$,
- $\gamma_i\gamma_j \leq \gamma_{i+k}\gamma_{j-k}$, if $0 < k < j - i$.

### 2.2 Error-free transformations

The development of some families of more stable algorithms, which are called *compensated algorithms*, is based on the paper [15] on error-free transformations (EFT). For a pair of floating-point numbers $a, b \in \mathbb{F}$, when no underflow occurs, there exists a floating-point number $y$ satisfying $a \circ b = x + y$, where $x = \mathrm{fl}(a \circ b)$ and $\circ \in \{+, -, \times\}$. Then the transformation $(a, b) \longrightarrow (x, y)$ is regarded as an EFT. For division, the corresponding EFT is constructed using the remainder, so its definition is slightly different (see below). The EFT algorithms of the sum, product and division of two floating-point numbers are the TwoSum algorithm [19], the TwoProd algorithm [20] and the DivRem algorithm [21, 22], respectively.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

### 2.3 Condition numbers

The condition number of polynomials is with respect to the difficulty of the evaluation algorithm. We assume to evaluate a bivariate polynomial $p(x, y)$ in basis $u \in \mathcal{U}$ at the point $(x, y)$, then for any $(x, y) \in I$, we have

$$\begin{aligned} |p(x,y) - \widehat{p}(x,y)| &= |\sum_{i=0}^{n}\sum_{j=0}^{m} c_{i,j} u_i^n(x) u_i^m(y)| \\ &\leq \sum_{i=0}^{n}\sum_{j=0}^{m} |c_{i,j}||u_i^n(x)||u_i^m(y)|. \end{aligned} \tag{4}$$

We assume that

$$\bar{p}(x,y) := \sum_{i=0}^{n}\sum_{j=0}^{m} |c_{i,j}||u_i^n(x)||u_i^m(y)|, \tag{5}$$

then the relative condition number is

$$cond(p,x,y) = \frac{\bar{p}(x,y)}{|p(x,y)|}. \tag{6}$$

In [23], it is known that the condition number in VS basis is as same as in Bernstein basis.

## 3 The compensated VS algorithm for Bézior tensor product surfaces

In this section, we show the VS algorithms, including univariate and bivariate ones. We provide a compensated VSTP algorithm for evaluating Bézior tensor product polynomials. Its forward error bound is also given in the end.

### 3.1 VS algorithm

The VS algorithm is a nested-type algorithm for the evaluation of bivariate polynomials of total degree $n$ by Schumaker and Volk [4]. Basically, the VS tensor product algorithm could be represented by the univariate VS algorithm.

Theorem 1 states the forward error bound of VS algorithm.

**Theorem 1.** [24] *Let $p(t) = \sum_{i=0}^{n} c_i z_i^n(t)$ with floating point coefficients $c_i$ and a floating point value $t$. Consider the computed result $\widehat{p}(t)$ with the VS algorithm and its corresponding theoretical result $p(t)$, if $4nu < 1$ where $u$ is the unit roundoff, then*

$$|p(t) - \widehat{p}(t)| \leq \gamma_{4n} \sum_{i=0}^{n} |c_i z_i^n(t)|. \tag{7}$$

Similar as Theorem 4 in [10], the forward error bound of the VSTP algorithm is easily performed in Theorem 2.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

**Algorithm 1** Volk-Schumaker algorithm [4] ($x \in [0,1]$)

function $res = \texttt{VS}(p, x)$
if $x \geq 1/2$
  $q = (1 \ominus x) \oslash x$
  $f = \texttt{Horner}((p_1, p_2, \ldots, p_n), q)$
  $res = f \otimes x^n$
else
  $q = x \oslash (1 \ominus x)$
  $f = \texttt{Horner}((p_{n-1}, p_{n-2}, \ldots, p_0), q)$
  $res = f \otimes (1 \ominus x)^n$
end

**Algorithm 2** VS tensor product algorithm

function $VSTP(p, x, y)$
for $i = n : -1 : 0$
  $\widehat{b}_{i,0} = VS(c_{i,:}, y)$
end
$\widehat{a}_0 = VS(\widehat{b}_{:,0}, x)$
$VSTP(p, x, y) \equiv \widehat{a}_0$

**Theorem 2.** *Let $p(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} c_{i,j} z_i^n(x) z_i^m(y)$ with floating point coefficients $c_{i,j}$ and floating point values $x, y$. Consider the computed result $\widehat{p}(x,y)$ of the VSTP algorithm and its corresponding theoretical result $p(x,y)$, if $(4n + 4m + 1)u < 1$ where $u$ is the unit roundoff, then*

$$|p(x,y) - \widehat{p}(x,y)| \leq \gamma_{4(n+m)+1}\bar{p}(x,y), \tag{8}$$

*where $\bar{p}(x,y)$ is defined in (5) in VS basis.*

### 3.2 The CompVSTP algorithm

The CompVS algorithm [23] is proposed by Delgado and Peña, which is as accurate as computing in twice the working precision by VS algorithm. In this section, in order to easily provide the forward error bound of CompVS algorithm, we show a compensated Horner algorithm with double-double precision input in Algorithm 3. A compensated power evaluation algorithm in Algorithm 4 is also given.

In Algorithm 3, assuming input $x$ is real number, and we split $x$ into three parts, i.e. $x = x^{(h)} + x^{(l)} + x^{(m)}$,where $x^{(h)}, x^{(l)} \in \mathbb{F}$, $x, x^{(m)} \in \mathbb{R}$ and $|x^{(l)}| \leq u|x^{(h)}|, |x^{(m)}| \leq u|x^{(l)}|$. Since the perturbation of input $x^{(m)}$ in Algorithm 3 is $\mathcal{O}(u^2)$, we just need to consider $x$ in double-double precision. According to Theorem 3.1 in [25], the proof of forward error bound of Algorithm 3 in the following theorem is similar as Theorem 12 in [11] .

**Theorem 3.** *If $p(x) = \sum_{i=0}^{n} a_i x^i$ $(n \geq 2)$ with floating point coefficients $a_i$ and a double-double precision number $x$. And $\widehat{\epsilon b}_0$ is the computed result err of the*

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

**Algorithm 3** Compensated Horner scheme with double-double precision inputs

function $[res, err] = \texttt{CompHorner2}(p, x^{(h)}, x^{(l)})$
$\widehat{b}_{n+1} = \widehat{\epsilon b}_{n+1} = 0$
for $i = n : -1 : 0$
  $[s_i, \pi_i] = \texttt{TwoProd}(\widehat{b}_{i+1}, x^{(h)})$
  $[\widehat{b}_i, \sigma_i] = \texttt{TwoSum}(s_i, a_i)$
  $\widehat{\epsilon b}_i = \widehat{\epsilon b}_{i+1} \otimes x^{(h)} \oplus \widehat{b}_{i+1} \otimes x^{(l)} \oplus \pi_i \oplus \sigma_i$
end
$[res, err] = [\widehat{b}_0, \widehat{\epsilon b}_0]$
$\texttt{CompHorner2}(p, x) \equiv \widehat{b}_0 \oplus \widehat{\epsilon b}_0$

*CompHorner2 algorithm, $\epsilon b_0$ is corresponding theoretical result of $\widehat{\epsilon b}_0$. Then*

$$|\epsilon b_0 - \widehat{\epsilon b}_0| \leq \gamma_{3n-1}\gamma_{3n} \sum_{i=0}^{n} |a_i||x^i|. \tag{9}$$

Graillat proposes a compensated power evaluation algorithm [26] as follows.

**Algorithm 4** Compensated power evaluation[26]

function $[res, err] = \texttt{CompLinPower}(x, n)$
$p_0 = x$
$e_0 = 0$
for $i = 1 : n - 1$
  $[p_i, \pi_i] = \texttt{TwoProd}(p_{i-1}, x)$
end
$[res, err] = [p_n, \texttt{Horner}((\pi_1, \pi_2, \ldots, \pi_{n-1}), x)]$
$\texttt{CompLinpower}(x, n) \equiv res \oplus err$

**Theorem 4.** [26] *If $p(x) = x^n$ $(n \geq 2)$ with a floating-point number $x$. And $\widehat{e}$ is the computed result err of the CompLinpower algorithm, e is corresponding theoretical result of $\widehat{e}$. Then*

$$|e - \widehat{e}| \leq \gamma_n\gamma_{2n}|x^n|. \tag{10}$$

In [23], Delgado and Peña present the running error analysis of CompVS algorithm, but they do not propose its forward error analysis. Here, combining Algorithms 3 and 4, we show the CompVS algorithm in the following algorithm which is expressed a little different in [23].

In Algorithm 5, we can easily obtain that $[q^{(h)}, q^{(l)}]$ is the double-double form of $q = (1 - x)/x$ if $x \geq 1/2$ or $q = x/(1 - x)$ if $x > 1/2$. Then, according to Theorems 1, 3 and 4, the forward error bound of CompVS algorithm is proposed in Theorem 5.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

**Algorithm 5** Compensated Volk-Schumaker algorithm ($x \in [0,1]$)

function $[res, err] = \texttt{CompVS}(\texttt{p}, \texttt{x})$
$[r, \rho] = \texttt{TwoSum}(1, -x)$
if $x \geq 1/2$
  $[q^{(h)}, \beta] = \texttt{DivRem}(r, x)$
  $q^{(l)} = (\rho \oplus \beta) \oslash x$
  $[f, e_1] = \texttt{CompHorner2}((p_1, p_2, \ldots, p_n), q^{(h)}, q^{(l)})$
  $[s, e_2] = \texttt{CompLinPower}(x, n)$
  $[res, err] = [f \otimes s, e_1 \otimes s \oplus e_2 \otimes f]$
else
  $[q^{(h)}, \beta] = \texttt{DivRem}(x, r)$
  $q^{(l)} = (\beta \ominus \rho \otimes q^{(h)}) \oslash r$
  $[f, e_1] = \texttt{CompHorner2}((p_{n-1}, p_{n-2}, \ldots, p_0), q^{(h)}, q^{(l)})$
  $[s, e_2] = \texttt{CompLinPower}(r, n)$
  $[res, err] = [f \otimes s, e_1 \otimes s \oplus e_2 \otimes f]$
end
$\texttt{CompVS}(x, n) \equiv res \oplus err$

**Theorem 5.** *If $p(t) = \sum_{i=0}^{n} c_i z_i^n(t)$ with floating point coefficients $c_i$ and a floating point value $t$. And $\widehat{\epsilon b_0}$ is the computed result $err$ of the CompVS algorithm, $\epsilon b_0$ is corresponding theoretical result of $\widehat{\epsilon b_0}$. Then*

$$|\epsilon b_0 - \widehat{\epsilon b_0}| \leq \gamma_{3n+1}\gamma_{3n+2} \sum_{i=0}^{n} |c_i z_i^n(t)|. \tag{11}$$

*Proof.* In Algorithm 5, we assume that $\widehat{f}+e_1 = \sum_{i=1}^{n} p_i q^i$ and $\widehat{s}+e_2 = x^n$ . Then, we can obtain that $p(t) = (\widehat{f}+e1)(\widehat{s}+e2)$ and assume that $e = e_1\widehat{s}+e_2\widehat{f}+e_1e_2$. Since $\widehat{e} = \widehat{e}_1 \otimes \widehat{s} \oplus \widehat{e}_2 \otimes \widehat{f}$, we have

$$\begin{aligned} &|e - \widehat{e}| \\ &\leq |(1+u)^2[(e_1 - \widehat{e}_1)\widehat{s} + (e_2 - \widehat{e}_2)\widehat{f} + e_1e_2] - (2u+u^2)e| \\ &\leq (2u+u^2)|e| + (1+u)^2(|e_1 - \widehat{e}_1||\widehat{s}| + |e_2 - \widehat{e}_2||\widehat{f}|). \end{aligned} \tag{12}$$

From Theorem 1, let $\bar{p}(t) = |c_i z_i^n(t)|$, we obtain that

$$|e| \leq \gamma_{4n}\bar{p}(t). \tag{13}$$

Thus

$$(2u+u^2)|e| \leq \gamma_2\gamma_{4n+1}\bar{p}(t). \tag{14}$$

According to Theorem 3, we have

$$(1+u)^2|e_1 - \widehat{e}_1||\widehat{s}| \leq \gamma_{3n}\gamma_{3n+1}\bar{p}(x) + \mathcal{O}(u^2). \tag{15}$$

According to Theorem 4, we have

$$(1+u)^2|e_2 - \widehat{e}_2||\widehat{f}| \leq \gamma_{n+1}\gamma_{2n+1}\bar{p}(x) + \mathcal{O}(u^2). \tag{16}$$

From (14) (15) and (16), we can deduce (11).

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

In fact, $p(x) = \widehat{p}(x) + \epsilon b_0$, where $\epsilon b_0$ is corresponding theoretical error of the computed result $\widehat{p}(x)$. In order to correct the result by Algorithm 1, Algorithm 5 find an approximate value $\widehat{\epsilon b}_0$ of $\epsilon b_0$. Motivated by this principle, we propose to use the CompVS algorithm instead of VS algorithm in Algorithm 2 to improve the accuracy of VSTP algorithm. According to Algorithm 2, we assume that

$$b_{i,0} = \widehat{b}_{i,0} + err_{i,0}^{(1)}, \quad 0 \le i \le n, \tag{17}$$

where $err_{i,0}^{(1)}$ is the theoretical error of $\widehat{b}_{i,0} = \texttt{VS}(c_{i,:}, y)$ and

$$b_{i,0} = \sum_{j=0}^{m} c_{i,j} z_i^m(y), \tag{18}$$

is the exact result for each $i$. Similarly, we have

$$\tilde{a}_0 = \widehat{a}_0 + err^{(2)}, \tag{19}$$

where $err^{(2)}$ is the theoretical error of $\widehat{a}_0 = \texttt{VS}(\widehat{b}_{:,0}, x)$ and

$$\tilde{a}_0 = \sum_{i=0}^{n} \widehat{b}_{i,0} z_i^n(x), \tag{20}$$

is the exact result. According to (17)-(20), we can deduce

$$\sum_{i=0}^{n}\sum_{j=0}^{m} c_{i,j} z_i^n(x) z_i^m(y) = \widehat{a}_0 + \sum_{i=0}^{n} err_{i,0}^{(1)} z_i^n(x) + err^{(2)}, \tag{21}$$

i.e.

$$p(x,y) = \widehat{p}(x,y) + \sum_{i=0}^{n} err_{i,0}^{(1)} z_i^n(x) + err^{(2)}. \tag{22}$$

Using CompVS algorithm, we can easily get the approximation values of $err_{i,0}^{(1)}$ and $err^{(2)}$, i.e. $\widehat{err}_{i,0}^{(1)}$ and $\widehat{err}^{(2)}$. Thus, we propose the CompVSTP algorithm for evaluating Bézier tensor product polynomials in Algorithm 6.

From (21) and (22), we assume that $e_1 = \sum_{i=0}^{n} err_{i,0}^{(1)} z_i^n(x)$ and $e_2 = err^{(2)}$ so that the real error of the computed result is $e = e_1 + e_2$, i.e. $p(x,y) = \widehat{p}(x,y) + e$. Firstly, we present the bound of $|e_1 - \widehat{e}_1|$ in Lemma 1.

**Lemma 1.** *From Algorithm 6, we assume that* $e_1 = \sum_{i=0}^{n} err_{i,0}^{(1)} z_i^n(x)$. *Then we have*

$$|e_1 - \widehat{e}_1| \le (\gamma_{3n+1}\gamma_{3n+2}(1+\gamma_{4m}) + \gamma_{4n}\gamma_{4m}\bar{p}(x,y), \tag{23}$$

*where* $\bar{p}(x,y)$ *is defined in (5) in VS basis.*

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

**Algorithm 6** Compensated VSTP algorithm ($x \in [0,1]$)

function $[res, err] = \texttt{CompVSTP}(p, x, y)$
$f_{i,j}^{(0)} = b_{i,j}$
for $i = 1 : m$
  $[\widehat{f}_{i,0}^{(1)}, \widehat{e}_{i,0}] = \texttt{CompVS}(f_{i,:}^{(0)}, y)$
end
$[\widehat{f}_{0,0}^{(2)}, \widehat{e2}] = \texttt{CompVS}(\widehat{f}_{:,0}^{(1)}, x)$
$[res, err] = [\widehat{f}_{0,0}^{(2)}, \widehat{e2} \oplus \texttt{VS}(\widehat{e1}_{:,0}, x)]$
$\texttt{CompVSTP}(p, x, y) \equiv res \oplus err$

*Proof.* We denote that

$$\bar{e}_1 = \sum_{i=0}^{n} \widehat{err}_{i,0}^{(1)} z_i^n(x). \tag{24}$$

Hence, we have

$$|e_1 - \widehat{e}_1| \le |e_1 - \bar{e}_1| + |\bar{e}_1 - \widehat{e}_1|. \tag{25}$$

According to Theorem 5, we have

$$|err_{i,0}^{(1)} - \widehat{err}_{i,0}^{(1)}| \le \gamma_{3n+1}\gamma_{3n+2} \sum_{j=0}^{m} |c_{i,j} z_i^m(y)|, \tag{26}$$

thus

$$\begin{aligned} |e_1 - \bar{e}_1| &= \sum_{i=0}^{n} |err_{i,0}^{(1)} - \widehat{err}_{i,0}^{(1)}| z_i^n(x) \\ &\le \gamma_{3n+1}\gamma_{3n+2} \sum_{i=0}^{n}\sum_{j=0}^{m} |c_{i,j} z_i^n(x) z_i^m(y)|. \end{aligned} \tag{27}$$

According to Theorem 1, we obtain

$$|\bar{e}_1 - \widehat{e}_1| \le \gamma_{4m} \sum_{i=0}^{n} |\widehat{err}_{i,0}^{(1)} z_i^n(x)|. \tag{28}$$

Then we have that

$$|\widehat{err}_{i,0}^{(1)}| \le |err_{i,0}^{(1)}| + |err_{i,0}^{(1)} - |\widehat{err}_{i,0}^{(1)}|. \tag{29}$$

By Theorem 1, we have

$$|err_{i,0}^{(1)}| \le \gamma_{4n} \sum_{j=0}^{m} |c_{i,j} z_i^m(y)|. \tag{30}$$

From (26), (29) and (30), we deduce that

$$|\widehat{err}_{i,0}^{(1)}| \le (\gamma_{3n+1}\gamma_{3n+2} + \gamma_{4n}) \sum_{j=0}^{m} |c_{i,j} z_i^m(y)|, \tag{31}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

and then from (28) we obtain

$$|\bar{e}_1 - \widehat{e}_1| \leq \gamma_{4m}(\gamma_{3n+1}\gamma_{3n+2} + \gamma_{4n})\bar{p}(x,y). \tag{32}$$

Hence, from (25), (27) and (32), we can obtain (23).

Then, we present the bound of $|e_2 - \widehat{e}_2|$ in Lemma 2.

**Lemma 2.** *From Algorithm 6, we assume that $e_2 = err^{(2)}$. Then we have*

$$|e_2 - \widehat{e}_2| \leq \gamma_{3m+1}\gamma_{3m+2}(1+\gamma_{4m})\bar{p}(x,y), \tag{33}$$

*where $\bar{p}(x,y)$ is defined in (5) in VS basis.*

*Proof.* According to Theorem 5, we have

$$|e_2 - \widehat{e}_2| \leq \gamma_{3m+1}\gamma_{3m+2}\sum_{i=0}^{n}|\widehat{b}_{i,0}z_i^n(x)|. \tag{34}$$

From Theorem 1, we obtain

$$|\widehat{b}_{i,0}| \leq \sum_{j=0}^{m}(1+\gamma_{4m})|c_{i,j}z_i^m(y)|. \tag{35}$$

Hence, from (34) and (35), we can deduce (33).

Above all, the forward error bound of CompVSTP algorithm is performed in the following theorem.

**Theorem 6.** *Let $p(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m}c_{i,j}z_i^n(x)z_i^m(y)$ with floating point coefficients $c_{i,j}$ and floating point values $x, y$. The forward error bound of Algorithm 6 is*

$$|CompVSTP(p,x,y) - p(x,y)| \leq u|p(x,y)| + 3(\gamma_{4n+2}^2 + \gamma_{4m+2}^2)\bar{p}(x,y), \tag{36}$$

*where $\bar{p}(x,y)$ is defined in (5) in VS basis.*

*Proof.* We assume that $e_1 = \sum_{i=0}^{n} err_{i,0}^{(1)}x^i$ and $e_2 = err^{(2)}$ so that $e = e_1 + e_2$. From (22), we have

$$p(x,y) = \widehat{p}(x,y) + e, \tag{37}$$

and from Algorithm 6, we have

$$\texttt{CompVSTP}(p,x,y) = \widehat{p}(x,y) \oplus \widehat{e}. \tag{38}$$

Hence

$$\begin{aligned} |\texttt{CompVSTP}(p,x,y) - p(x,y)| &\leq |(1+u)(p(x,y) - e + \widehat{e}) - p(x,y)| \\ &\leq u|p(x,y)| + (1+u)|e - \widehat{e}|. \end{aligned} \tag{39}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

Since $\widehat{e} = \widehat{e}_1 \oplus \widehat{e}_2$, we have

$$\begin{aligned} |e - \widehat{e}| &\leq |(1+u)(e_1 - \widehat{e}_1 + e_2 - \widehat{e}_2) - ue| \\ &\leq u|e| + (1+u)(|e_1 - \widehat{e}_1| + |e_2 - \widehat{e}_2|). \end{aligned} \tag{40}$$

From Theorem 2, we obtain that

$$|e| \leq \gamma_{4(n+m)+1}\bar{p}(x,y). \tag{41}$$

Thus

$$u(1+u)|e| \leq \gamma_1\gamma_{4(n+m+1)}\bar{p}(x,y) \leq \gamma_{4n+2}\gamma_{4m+2}\bar{p}(x,y) \leq \frac{1}{2}(\gamma_{4n+2}^2+\gamma_{4m+2}^2)\bar{p}(x,y). \tag{42}$$

According to Lemma 1, we have

$$\begin{aligned} (1+u)^2|e_1 - \widehat{e}_1| &\leq (2\gamma_{4n+1}^2 + \gamma_{4n+1}\gamma_{4m+1})\bar{p}(x,y) \\ &\leq (\frac{5}{2}\gamma_{4n+1}^2 + \frac{1}{2}\gamma_{4m+1}^2)\bar{p}(x,y). \end{aligned} \tag{43}$$

According to Lemma 2, we have

$$(1+u)^2|e_2 - \widehat{e}_2| \leq 2\gamma_{4m+1}^2\bar{p}(x,y). \tag{44}$$

From (42) (43) and (44), we can deduce (36).

According to the relative condition number defined in (6), we can deduce Corollary 1.

**Corollary 1.** *Let $p(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} c_{i,j}z_i^n(x)z_i^m(y)$ with floating point coefficients $c_{i,j}$ and floating point values $x, y$. The forward relative error bound of Algorithm 6 is*

$$\frac{|CompVSTP(p,x,y) - p(x,y)|}{|p(x,y)|} \leq u + 3(\gamma_{4n+2}^2 + \gamma_{4m+2}^2)cond(p,x,y). \tag{45}$$

## 4 Numerical experiments

In this section, we compare CompVSTP algorithm against an implementation of VSTP algorithm that applies the double-double format [14, 27] which we denote as DDVSTP algorithm. In fact, since the working precision is double precision, the double-double arithmetic is the most efficient way to yield a full precision accuracy of evaluating polynomials. Moreover, we also compare CompVSTP algorithm against compensated de Casteljau (CompDCTP) algorithm [10].

All our experiments are performed using IEEE-754 double precision as working precision. All the programs about accuracy measurements have been written in Matlab R2014a on a 1.4-GHz Intel Core i5 Macbook Air. We focus on the relative forward error bounds for ill-conditioned Bézier tensor product polynomials. We use a similar GenPoly algorithm [10, 21] to generate tested polynomials

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

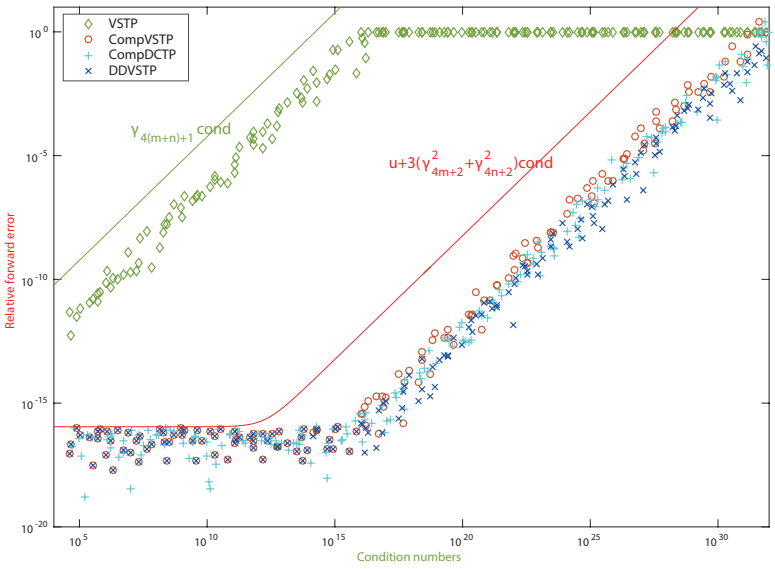


**Fig. 1.** Accuracy of evaluation of ill-conditioned Bézier tensor product polynomials with respect to the condition number

$p(x, y)$. The generated polynomials are $6 \times 7$ degree with condition numbers varying from $10^4$ to $10^{36}$, $x$ and $y$ are random numbers in $[0, 1]$ and the inspired computed results of all the tested polynomials are 1. We evaluate the polynomials by the VSTP, CompVSTP, CompDCTP, DDVSTP algorithms and the Symbolic Toolbox, respectively, so that the relative forward errors can be obtained by $(|p_{res}(x, y) - p_{sym}(x, y)|)/|p_{sym}(x, y)|$ and the relative error bounds are described from Corollary 1. Note that the condition number of Bézier tensor product polynomials in Bernstein basis evaluated by CompDCTP algorithm is as same as in VS basis evaluated by CompVSTP algorithm. Then we present the relative forward errors of evaluation of the tested polynomials in Figure 1. As we can see, the relative errors of CompVSTP, CompDCTP and DDVSTP algorithms are both smaller than $u$ ($u \approx 1.16 \times 10^{-16}$) when the condition number is less than $10^{16}$. And the accuracy of them is decreasing linearly for the condition number larger than $10^{16}$. However, the VSTP algorithm can not yield the working precision; the accuracy of which decreases linearly since the condition number is less than $10^{16}$.

At last, we give the computational cost of VSTP, CompVSTP, CompDCTP and DDVSTP algorithms.

- **VSTP**: (3n+2)(m+1)+3m+2 flops,
- **CompVSTP**: (50n+26)(m+1)+50m+26+1 flops,
- **CompDCTP**: $(24n^2$+24n+7)(m+1)+$24m^2$+24m+7+1 flops,

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

– **DDVSTP**: (68n+120)(m+1)+68m+120 flops.

CompVSTP and DDVSTP algorithms require almost 17 and 23 times flop than VSTP algorithm, respectively. Meanwhile, CompDCTP algorithm requires $\mathcal{O}(n^2m)$ flop which is much more than $\mathcal{O}(nm)$. Hence, CompVSTP algorithm only needs about 73.5% of flops counting on average of DDVSTP algorithm and needs much less computational cost than CompDCTP algorithm. Meanwhile, CompVSTP algorithm is as accurate as CompDCTP and DDVSTP algorithms.

## 5 Conclusions and further work

In this paper, we present CompVSTP algorithm to evaluate Bézier tensor product polynomials, which are compensated algorithms that obtaining an approximate error to correct the computed results by original algorithm. The proposed algorithm is as accurate as computing in double-double arithmetic which is the most efficient way to yield a full precision accuracy. Moreover, it needs fewer flops than counting on average with double-double arithmetic.

A similar approach can be applied to other problems to obtain compensated algorithms. For example we can consider the evaluation of ill-conditioned tensor product polynomials in orthogonal basis like Chebyshev and Legendre basis. Instead of tensor product surfaces, we can consider triangle surfaces like Bernstein-Bézier form. We can also study compensated algorithms for multivariate polynomials.

## References

1. G. Farin, Curves and Surfaces for Computer Aided Geometric Design, Academic Press, Inc., SanDiego, CA, fourth edition, 1997.
2. E. Mainar, J. Peña, Error analysis of corner cutting algorithms, Numerical Algorithms. Vol. 22 (1), pp. 41-52, 1999.
3. R. Barrio, A unified rounding error bound for polynomial evaluation, Advances in Computational Mathematics, Vol. 19 (4), pp. 385-399, 2003.
4. L. Schumaker, W. Volk, Efficient evaluation of multivariate polynomials. Comput. Aided Geom. Design Vol. 3, pp.149-154, 1986.
5. S. Graillat, P. Langlois, N.Louvet, Compensated Horner scheme, Technical Report, University of Perpignan, France, 2005.
6. S. Graillat, P. Langlois, N.Louvet, Algorithms for accurate, validated and fast polynomial evaluation, Japan J.Indust.Appl.Math, Vol. 26, pp. 191-214, 2009.
7. P. Langlois, N. Louvet, How to ensure a faithful polynomial evaluation with the compensated Horner algorithm, Proceedings 18th IEEE Symposium on Computer Arithmetic, IEEE Computer Society, pp. 141-149.
8. H. Jiang, S. G. Li, L. Z. Cheng, F. Su, Accurate evaluation of a polynomial and its derivative in Bernstein form, Comput.Math.Appl., Vol.60(3), pp.744-755, 2010.
9. H. Jiang, R. Barrio, X. K. Liao, L. Z. Cheng, Accurate evalution algorithm for bivariate polynomial in Bernstein-Bźier form, Appl.Numer.Math., Vol.61, pp.1147-1160, 2011.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6

10. H. Jiang, H. S. Li, L. Z. Cheng, R. Barrio, C. B. Hu, X. K. Liao, Accurate, Validated and Fast Evaluation of Bézier Tensor Product Surfaces, Reliable Computing, Vol.18, pp.55-72, 2013.
11. P. B. Du, H. Jiang, L. Z. Cheng, Accurate evaluation of polynomials in Legendre basis, Journal of Applied Mathematics Vol. 2014, Article ID 742538, 2014.
12. P. B. Du, H. Jiang, H. S. Li, L. Z. Cheng, C. Q. Yang, Accurate evaluation of bivariate polynomials, 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 51-55, 2016.
13. P. B. Du, R. Barrio, H. Jiang, L. Z. Cheng, Accurate Quotient-Difference algorithm: error analysis, improvements and applications, Applied Mathematics and Computation, Vol. 309, pp. 245-271, 2017.
14. X. S. Li, J. W. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, A. Kapur, M. C. Martin, T. Tung, D. J. Yoo, Design, implementation and testing of extended and mixed precision BLAS, ACM Trans.MATH.Software, Vol. 28(2), pp. 152-205, 2002.
15. T. Ogita, S. Rump, S. Oishi, Accurate sum and dot product, SIAM J.Sci.Comput, Vol. 26, pp. 1955-1988, 2005.
16. S. Rump, T. Ogita, S. Oishi, Accurate floating-point summation part I: Faithful rounding, SIAM J.Sci.Comput, Vol. 31, pp. 189-224, 2008.
17. S. Rump, T. Ogita, S. Oishi, Accurate floating-point summation part II: Sign, $k$-fold faithful and rounding to nearest, SIAM J.Sci.Comput, Vol.31, pp. 1269-1302, 2008.
18. N. J. Higham, Accuracy and Stability of Numerical Algorithm, second ed., SIAM, Philadelphia, 2002.
19. D. E. Knuth, The Art of Computer Programming: Seminumerical Algorithms, third ed., Addison-Wesley, 1998.
20. T. J. Dekker, A floating-point technique for extending the available precision, Numer.Math, Vol.18, pp. 224-242, 1971.
21. N. Louvet, Compensated algorithms in floating-point arithmetic: accuracy, validation, performances, Ph.D. thesis, Université de Perpignan Via Domitia, 2007.
22. M. Pichat, J. Vignes, Ingénierie du contrôle de la préision des calculs sur ordinateur, Tech. rep., Editions Technip, 1993.
23. J. Delgado, J. Peña, Algorithm 960: POLYNOMIAL: An Object-Oriented Matlab Library of Fast and Efficient Algorithms for Polynomials. ACM Trans.MATH.Software, Vol. 42, 3, Article 23, 19 pages, 2016.
24. J. Delgado, J. Peña, Running relative error for the evaluation of polynomials. SIAM J. Sci. Comput. Vol. 31, pp. 3905-3921, 2009.
25. J. Peña, T. Sauer, On the multivariate Horner scheme, SIAM J.Numer.Anal, Vol. 37, No. 4, pp. 1186-1197, 2000.
26. S. Graillat, Accurate Floating Point Product and Exponentiation, IEEE Transactions on Computers, Vol. 58 (7), pp. 994-1000, 2009.
27. Y. Hida, X. Y. Li, D. H. Bailey, Algorithms for Quad-Double Precision Floating Point Arithmetic, 15th IEEE Symposium on Computer Arithmetic, IEEE Computer Society, pp.155-162, 2001.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_6