mathematics

MDPI

Article

Compensated Evaluation of Tensor Product Surfaces in CAGD

Jorge Delgado Gracia

Departamento de Matemática Aplicada, Universidad de Zaragoza, 44003-Teruel, Spain; jorgedel@unizar.es; Tel.: +34-978618174

Received: 15 November 2020; Accepted: 7 December 2020; Published: 14 December 2020

Abstract: In computer-aided geometric design, a polynomial surface is usually represented in Bézier form. The usual form of evaluating such a surface is by using an extension of the de Casteljau algorithm. Using error-free transformations, a compensated version of this algorithm is presented, which improves the usual algorithm in terms of accuracy. A forward error analysis illustrating this fact is developed.

Keywords: Bernstein basis; polynomial algorithms; tensor product surfaces; error analysis; error-free transformations

1. Introduction

The Horner algorithm is the most usual method for the evaluation of polynomials. Important algorithms in computer-aided geometric design (CAGD) need to compute roots of curves and surfaces. Some of the algorithms, in order to compute those roots, need to evaluate accurately the curves and surfaces at points close to the roots (see [1,2]). These evaluations are ill-conditioned, and accurate evaluation algorithms could play a key role in the performances of some of these root finding algorithms. In the last few years, in the literature it has been shown that the de Casteljau algorithm outperforms Horner's algorithm, among other evaluation algorithms, from the point of view of accuracy (see [3–8]). The de Casteljau algorithm evaluates polynomials represented in Bézier form, that is, using the Bernstein polynomials. In CAGD it is the usual evaluation algorithm for polynomial curves.

In CAGD, polynomials (curves and surfaces) are usually represented in Bernstein form, by using the Bernstein polynomials of degree n. A polynomial in the Bézier form is evaluated by the de Casteljau algorithm in the bivariate case and by an extended version in the multivariate case. The error analysis of these algorithms in [6,7] shows a relative error bound of the following form:

$$\text{Condition number} \times \mathcal{O}(u), \tag{1}$$

where u is the unit roundoff of the computing precision. For an ill-conditioned problem, such as the evaluation of a polynomial at parameters very close to a multiple root, the condition number can exceed $1/u$. In that case we can obtain an approximation of the polynomial at the parameter value with almost all its digits being false.

Error-free transformations (EFTs) have been studied by Rump and Ogita in [9–11]. In [12], applying EFTs, Graillat and Langlois presented a compensated version of the usual Horner algorithm to evaluate polynomials represented in the power basis. Later, in [13] a compensated de Casteljau algorithm for the evaluation of univariate polynomialas was devised. The relative error bound for this algorithm has the following form:

$$u + \text{Condition number} \times \mathcal{O}(u^2),$$

which improves the bound for the usual de Casteljau algorithms in (1).

In [7], an error analysis was performed for the extension of the de Casteljau algorithm for tensor product surfaces in Bernstein-Bézier form. In this paper, applying EFTs, we present a compensated version of this algorithm for the evaluation of those surfaces with improved accuracy.

The layout of the paper is as follows. Section 2 introduces some basic notation and results about error analysis with floating point arithmetic; the EFTs; the de Casteljau algorithm for polynomial curves and its compensated version. Section 3 recalls the extension of the de Casteljau algorithm for the evaluation of tensor product surfaces and the corresponding error analysis. Then, the compensated de Casteljau algorithm for Bézier tensor product surfaces is devised and the corresponding error analysis performed, providing a better bound for the error.

2. Basic Notation and Results

2.1. Floating Point Arithmetic and Forward Error Analysis

Given a real number x, the computed element in floating point arithmetic will be denoted by either $fl(x)$ or $\widehat{x}$. Let us assume that u is the unit roundoff of the arithmetic floating point system we are using. In error analysis, the study of the effect of rounding errors is usually carried out by using one of the following two models.

$$fl(a\ op\ b) = (a\ op\ b)\,(1+\delta) \quad \text{or} \quad fl(a\ op\ b) = \frac{a\ op\ b}{1+\delta}, \quad |\delta| \le u, \tag{2}$$

where op is any one of the operations $+, -, \times, /$ (for more details see pages 40–41 of [14]). Now let us define

$$\gamma_k := \frac{ku}{1-ku} = ku + \mathcal{O}(u^2), \tag{3}$$

where $k \in \mathbb{N}_0$ verifies $ku < 1$. Given $\delta_1, \ldots, \delta_k$ with $|\delta_i| \le u$ for all i, in error analysis it is usual to deal with quantities θ_k satisfying that $\prod_{i=1}^{k}(1+\delta_i) = (1+\theta_k)$. In Lemma 3.1 of [14] it was proved that their absolute value is bounded above by γ_k, that is, $|\theta_k| \le \gamma_k$. The following result summarizes some classic properties in error analysis (see Lemma 3.3 of [14]).

Lemma 1.

i. $(1+\theta_k)(1+\theta_j) = 1+\theta_{k+j}$,
ii. $\gamma_k\gamma_j \le \gamma_{\min(k,j)}$ *for* $\max(j,k)u \le 1/2$,
iii. $i\gamma_k \le \gamma_{ki}$,
iv. $\gamma_k + u \le \gamma_{k+1}$,
v. $\gamma_k + \gamma_j + \gamma_k\gamma_j \le \gamma_{k+j}$.

Condition numbers of the functions to be evaluated are important for the accuracy of the result. Let us now recall some condition numbers related to the evaluation of functions. Given a space of functions $\mathcal{U}$ defined on $\Theta \subset \mathbb{R}^s$, a basis $B = (b_0, \ldots, b_n)$ for $\mathcal{U}$ and a function $f = \sum_{i=0}^{n} c_i b_i \in \mathcal{U}$, measures of the sensitivity of $f(x)$ to perturbations in $c = (c_j)_{j=0}^{n}$ are important in error analysis of the evaluation algorithms. Thus, given a relative perturbation $\delta = (\delta_i)_{i=0}^{n}$ of the coefficients c, we obtain the function $g = \sum_{i=0}^{n}(1+\delta_i)c_i b_i$, which is related to f. Then for any $x \in \Theta$

$$|f(x) - g(x)| = \left|\sum_{i=0}^{n} \delta_i c_i b_i(x)\right| \le \|\delta\|_\infty \sum_{i=0}^{n} |c_i b_i(x)|. \tag{4}$$

The number

$$S_B(f(x)) := \sum_{i=0}^{n} |c_i b_i(x)|,$$

plays the role of a condition number for the evaluation of the function f at x using the basis B (see [4,5,15–17]).

In CAGD, it is usual that the basis B must be formed of blending functions; that is, each basis function must be nonnegative on Θ, and the sum of all bases functions must be equal to 1 for all point in Θ. If $B = (b_0, \ldots, b_n)$ is a basis of blending functions and $\bar{B} = (k_0 b_0, \ldots, k_n b_n)$ ($k_i \in \mathbb{R}\ \forall\, i$) then $S_B(f(x)) = S_{\bar{B}}(f(x))$.

In floating point arithmetic, given an algorithm for the evaluation of the function $f(x)$, one obtains the computed value $fl(f(x))$ or $\widehat{f(x)}$. From a practical point of view, to obtain an error bound or estimate for the approximation of the exact evaluation $f(x)$ given by $fl(f(x))$, it is desirable. The success on the accuracy of the obtained aproximation when using an evaluation algorithm despends on:

- The backward error—that is, the error of the calculations of the algorithm;
- The difficulty of the evaluated function—that is, the condition number of the function with respect to the basis used as a representation by the evaluation algorithm.

In error analysis, the computed $fl(f(x))$ can be expressed as $fl(f(x)) = g(x) = \sum_{i=0}^{n}(1 + \delta_i)c_i b_i(x)$, where $\delta = (\delta_i)_{i=0}^{n}$ is a perturbation in c. Thus, the upper bound of the forward error for evaluation in formula (4) is usually interpreted as a product of the backward error $\|\delta\|_\infty$ and the condition number $S_B(f(x))$ (cf. [14]).

2.2. Error-Free Transformations

Error-free transformations (EFTs) will be taken into account in our algorithms in order to improve accuracy. In particular, `TwoSum` and `TwoProduct` EFTs will be used (see [9]) for computing sums and products, respectively. The algorithm `TwoSum` for the sum was presented by Knuth in [18], whereas the algorithm `TwoProduct` for the product was presented by Dekker, due to G. W. Veltkamp, in [19]. Algorithms 1 and 2 show these algorithms (`TwoSum` and `TwoProduct`), Algorithm 3 is used by Algorithm 2.

Algorithm 1 TwoSum algorithm.

Require: a, b
Ensure: $[x, y]$ such that $x + y = a + b$
$x = a \oplus b$
$z = x \ominus a$
$y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$

Algorithm 2 TwoProduct algorithm.

Require: a, b
Ensure: $[x, y]$ such that $x + y = a \cdot b$
1: $x = a \otimes b$
2: $[a_1, a_2] = Split(a)$
3: $[b_1, b_2] = Split(b)$
4: $y = a_2 \otimes b_2 \ominus (((x \ominus a_1 \otimes b_1) \ominus a_2 \otimes b_1) \ominus a_1 \otimes b_2)$

Algorithm 3 Split algorithm.

Require: a
Ensure: $[x, y]$ such that $x + y = a$
1: $c = \texttt{factor} \otimes a$ %`factor` $= 2^{27} + 1$ in IEEE 754
2: $x = c \ominus (c \ominus a)$
3: $y = a \ominus x$

Error analyses of both algorithms were presented in Theorem 3.4 of [9] and Théorème 3.14 of [20]. The following result shows a summary of these results.

Theorem 1. *Let $\mathbb{F}$ be the set of standard floating point numbers corresponding to a certain floating point arithmetic. If $a, b \in \mathbb{F}$, then:*

i. $[x, y] = \texttt{TwoSum}(a, b)$ *verifies*

$$a + b = x + y, \quad x = a \oplus b, \quad |y| \leq u|x|, \quad |y| \leq u|a + b|.$$

ii. $[x, y] = \texttt{TwoProduct}(a, b)$*; if not, underflow occurs,*

$$a \cdot b = x + y, \quad x = a \otimes b, \quad |y| \leq u|x|, \quad |y| \leq u|a \cdot b|.$$

2.3. De Casteljau Algorithm for Polynomial Curves in Bézier Form

The Horner algorithm is the best well-known method for polynomial evaluation. It uses the monomial basis of the space $\mathcal{P}_n$, $M^n := (m_0^n(t), m_1^n(t), \ldots, m_n^n(t))$, $t \in [0, 1]$, given by $m_i^n(t) = t^i$, $i = 0, 1, \ldots, n$. Given $p(t) = \sum_{i=0}^{n} c_i\, m_i^n(t)$, the error analysis of the Horner algorithm in chapter 5 of [14] shows that

$$|p(t) - fl(p(t))| \leq \gamma_{2n} \sum_{i=0}^{n} |c_i| t^i = \gamma_{2n}\, S_{M^n}(p(t)), \quad \text{for all } p \in \mathcal{P}_n \text{ and } t \in [0, 1].$$

In CAGD the usual evaluation algorithm for polynomial curves is the de Casteljau algorithm. This algorithm evaluates polynomials represented using the Bernstein basis (see [21]). The Bernstein polynomials of degree n, $B^n := (b_0^n(t), b_1^n(t), \ldots, b_n^n(t))$, $t \in [0, 1]$, form a basis of $\mathcal{P}_n$ and are defined by

$$b_i^n(t) = \binom{n}{i} t^i (1 - t)^{n-i}, \quad i = 0, 1, \ldots, n. \tag{5}$$

A polynomial

$$p(t) = \sum_{i=0}^{n} c_i\, b_i^n(t) \in \mathcal{P}_n \tag{6}$$

is said to be in Bézier form or in Bernstein–/Bézier form. Algorithm 4 shows the de Casteljau algorithm for the evaluation of polynomials in Bézier form (6).

Algorithm 4 De Casteljau algorithm for the evaluation of $p \in \mathcal{P}_n$ at t.

Require: $t \in [0, 1]$ and $(c_i)_{i=0}^n$
Ensure: $f_0^n(t) \approx \sum_{i=0}^{n} c_i b_i^n(t)$
  for $j = 0$ to n **do**
    $f_j^0(t) := c_j$
  end for
  for $r = 1$ to n **do**
    for $j = 0$ to $n - r$ **do**
      $f_j^r(t) = (1 - t) \otimes f_j^{r-1}(t) \oplus t \otimes f_{j+1}^{r-1}(t)$
    end for
  end for

A corner cutting algorithm is an algorithm such that each step is formed by linear convex combinations (see [6]). The de Casteljau algorithm is a corner cutting algorithm. In [6] an error analysis

of corner cutting algorithms was carried out, which for the particular case of the de Casteljau algorithm can be written as

$$|p(t) - fl(p(t))| \leq \gamma_{2n} \sum_{i=0}^{n} |c_i| b_i^n(t) = \gamma_{2n}\, S_{B^n}(p(t)), \quad \text{for all } p \in \mathcal{P}_n \text{ and } t \in [0,1].$$

In addition, the optimal conditioning of Bernstein basis for polynomial evaluation among all the bases formed by nonnegative polynomials on $[0,1]$ was shown in [5]. Thus, there does not exist another basis of $\mathcal{P}_n$, up to positive scaling, formed by nonnegative polynomials on $[0,1]$ that is better conditioned for every $p \in \mathcal{P}_n$ at every point $t \in [0,1]$. In particular, we have $S_{B^n}(p(t)) \leq S_{M^n}(p(t))$ for all $p \in \mathcal{P}_n$ and $t \in [0,1]$. Hence, the part of the error bound corresponding to the condition number for the de Casteljau algorithm is lower than the corresponding part of the bound for the Horner algorithm. In fact, the numerical experiments in [3] show that the algorithms using the Bernstein representation, like the de Casteljau algorithm, present better stability properties than the Horner algorithm.

2.4. Compensated Evaluation Algorithms for Bézier Curves

It is usual to apply EFTs (see [9] and Section 2.2) in order to devise compensated evaluation algorithms providing more accurate results. Hence, in [22,23] Graillat, Langlois and Louvet devised a compensated Horner algorithm for the evaluation of a polynomial in monomial form. In Theorem 5 of [22] it was proved that the evaluation of a degree n polynomial with the compensated Horner algorithm provides an approximation $fl(p(t))$ verifying

$$|p(t) - fl(p(t))| \leq u|p(t)| + \gamma_{2n}^2 S_{Mn}(p(t)).$$

In [13] a compensated de Casteljau algorithm for the evaluation of polynomials curves in Bernstein-Bézier form was presented. In Theorem 5 of [13] it was proved that the evaluation of a degree n polynomial with the compensated de Casteljau algorithm provides an approximation $fl(p(t))$ verifying

$$|p(t) - fl(p(t))| \leq u|p(t)| + 2\gamma_{3n}^2 S_{B^n}(p(t)).$$

According to the previous bound for problems where

$$2\gamma_{3n}^2 \frac{S_{B^n}(p(t))}{|p(t)|} < u,$$

the relative error for the approximations provided by the compensated de Casteljau algorithm is u.

3. Evaluation Algorithms for Tensor Product Bézier Surfaces

In CAGD ensor product polynomial surfaces are usually represented in the Bernstein-Bézier form (see [21]) by using tensor product Bernstein systems.

Definition 1. *Let $B^m = (b_0^m, \ldots, b_m^m)$ and $B^n = (b_0^n, \ldots, b_n^n)$ be two Bernstein systems defined on $[0,1]$, where b_i^k and $i = 0, 1 \ldots, k$, are the Bernstein polynomials of degree k. The system $B^m \otimes B^n := (b_i^m(x) \otimes b_j^n(y))_{i=0,\ldots,m}^{j=0,\ldots,n}$ is called a tensor product Bernstein system and the surface*

$$F(x,y) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij}\, b_i^m(x)\, b_j^n(y), \quad (x,y) \in [0,1] \times [0,1], \tag{7}$$

is called a tensor product Bézier surface.

A tensor product Bézier surface can be evaluated by de Casteljau type evaluation algorithm inspired in the de Castaljau evaluation algorithm for Bézier curves (see [21]). By considering the

components of the points P_{ij}, the evaluation of (7) depends on the evaluation of scalar functions. Hence, based on the de Casteljau algorithm for Bézier curves, the corresponding evaluation algorithm for tensor product Bézier surfaces is shown in Algorithm 5.

Algorithm 5 De Casteljau algorithm for the evaluation of F in (7).

Require: $(x,y) \in [0,1] \times [0,1]$ and $(f_{ij})_{i=0,j=0}^{m,n}$
Ensure: $f_{00}^{mn}(x,y) \approx \sum_{i=0}^{m}\sum_{j=0}^{n} f_{ij} b_i^m(x) b_j^n(y)$
for $i = 0$ to m **do**
for $j = 0$ to n **do**
$f_{ij}^{00}(x,y) = f_{ij}$
end for
end for
for $i = 0$ to m **do**
for $s = 1$ to n **do**
for $j = 0$ to $n - s$ **do**
$f_{ij}^{0s}(x,y) = (1-y)\, f_{ij}^{0,s-1}(x,y) + y\, f_{i,j+1}^{0,s-1}(x,y)$
end for
end for
end for
for $r = 1 : m$ **do**
for $i = 0$ to $m - r$ **do**
$f_{i0}^{rn}(x,y) = (1-x)\, f_{i0}^{r-1,n}(x,y) + x\, f_{i+1,0}^{r-1,n}(x,y)$
end for
end for

In Theorem 5 of [7], error analyses of algorithms evaluating tensor product surfaces were performed. Taking into account the roundoff error when computing $1 \ominus t$, for the particular case of tensor product Bézier surfaces we have the following error analysis of Algorithm 5.

Theorem 2. *Let us consider the system of functions* $B^{mn} = (b_i^m \otimes b_j^n)_{i=0,\dots,m}^{j=0,\dots,n}$ *defined on* $[0,1] \times [0,1]$. *Let* $F(x,y)$ *be given by* (7), *and let us suppose that* $3(m+n)u < 1$, *where* u *is the unit roundoff. Then, the value* $\widehat{F}(x,y) = \widehat{f}_{00}^{mn}$ *computed in floating point arithmetic through Algorithm 5 satisfies*

$$|F(x,y) - \widehat{F}(x,y)| \leq \gamma_{3(m+n)} S_{B^{mn}}(F(x,y)).$$

Compensated de Casteljau Evaluation Algorithm for Tensor Product Bézier Surfaces

In this section we devise a compensated de Casteljau algorithm for the evaluation of tensor product surfaces—that is, a compensated version of Algorithm 5. In order to track the local errors at each step, the following EFTs will be used:

$$[\widehat{r}_y, \rho_y] = TwoSum(1,-y), \qquad [\widehat{r}_x, \rho_x] = TwoSum(1,-x),$$
$$[P_{1,y}, \pi_{ij}^{0s}] = TwoProduct(\widehat{r}_y, \widehat{f}_{ij}^{0,s-1}), \qquad [P_{1,x}, \pi_{i0}^{rn}] = TwoProduct(\widehat{r}_x, \widehat{f}_{i0}^{r-1,n}),$$
$$[P_{2,y}, \sigma_{ij}^{0s}] = TwoProduct(y, \widehat{f}_{i,j+1}^{0,s-1}), \qquad [P_{2,x}, \sigma_{i0}^{rn}] = TwoProduct(x, \widehat{f}_{i+1,0}^{r-1,n}),$$
$$[\widehat{f}_{ij}^{0s}, \xi_{ij}^{0s}] = TwoSum(P_{1,y}, P_{2,y}), \qquad [\widehat{f}_{i0}^{rn}, \xi_{i0}^{rn}] = TwoSum(P_{1,x}, P_{2,x}).$$

Then we can describe the error in the following way:

$$l_{ij}^{0s} = \pi_{ij}^{0s} + \sigma_{ij}^{0s} + \xi_{ij}^{0s} + \rho_y \cdot \widehat{f}_{ij}^{0,s-1}, \qquad l_{i0}^{rn} = \pi_{i0}^{rn} + \sigma_{i0}^{rn} + \xi_{i0}^{rn} + \rho_x \cdot \widehat{f}_{i0}^{r-1,n},$$
$$(1-y) \cdot \widehat{f}_{ij}^{0,s-1} + y \cdot \widehat{f}_{i,j+1}^{0,s-1} = \widehat{f}_{ij}^{0s} + l_{ij}^{0s}, \qquad (1-x) \cdot \widehat{f}_{i0}^{r-1,n} + x \cdot \widehat{f}_{i+1,0}^{r-1,n} = \widehat{f}_{i0}^{rn} + l_{i0}^{rn}.$$

Now let us define the global errors at each step as

$$\partial f_{ij}^{0s} = f_{ij}^{0s} - \widehat{f}_{ij}^{0s}, \quad \partial f_{i0}^{rn} = f_{i0}^{rn} - \widehat{f}_{i0}^{rn}.$$

It can be seen that the local error satisfies the following expressions:

$$\partial f_{ij}^{0s} = (1-y) \cdot \partial f_{ij}^{0,s-1} + y \cdot \partial f_{i,j+1}^{0,s-1} + l_{ij}^{0s}, \quad \partial f_{i0}^{rn} = (1-x) \cdot \partial f_{i0}^{r-1,n} + x \cdot \partial f_{i+1,0}^{r-1,n} + l_{i0}^{rn}. \tag{8}$$

If computations are performed in exact arithmetic:

$$F(x,y) = \widehat{f}_{00}^{mn} + \partial f_{00}^{mn}. \tag{9}$$

Taking into account the previous discussion, Algorithm 6 shows the corresponding compensated version of the de Casteljau algorithms for tensor product Bézier surfaces.

Algorithm 6 Compensated de Casteljau algorithm for the evaluation of F in (7).

Require: $(x,y) \in [0,1] \times [0,1]$ and $(f_{ij})_{i=0,j=0}^{m,n}$
Ensure: $res \approx \sum_{i=0}^{m} \sum_{j=0}^{n} f_{ij} b_i^m(x) b_j^n(y)$
  $[\widehat{r}_y, \rho_y] = TwoSum(1, -y)$
  for $i = 0$ to m **do**
    for $j = 0$ to n **do**
      $\widehat{f}_{ij}^{00}(x,y) = f_{ij}$
      $\widehat{\partial f}_{ij}^{00}(x,y) = 0$
    end for
  end for
  for $i = 0$ to m **do**
    for $s = 1$ to n **do**
      for $j = 0$ to $n - s$ **do**
        $[P_{1,y}, \pi_{ij}^{0s}] = TwoProduct(\widehat{r}_y, \widehat{f}_{ij}^{0,s-1}(x,y))$
        $[P_{2,y}, \sigma_{ij}^{0s}] = TwoProduct(y, \widehat{f}_{i,j+1}^{0,s-1}(x,y))$
        $[\widehat{f}_{ij}^{0s}(x,y), \xi_{ij}^{0s}] = TwoSum(P_{1,y}, P_{2,y})$
        $\widehat{l}_{ij}^{0s} = \pi_{ij}^{0s} \oplus \sigma_{ij}^{0s} \oplus \xi_{ij}^{0s} \oplus \rho_y \otimes \widehat{f}_{ij}^{0,s-1}(x,y)$
        $\widehat{\partial f}_{ij}^{0s}(x,y) = \widehat{l}_{ij}^{0s} \oplus \left(\widehat{r}_y \otimes \widehat{\partial f}_{ij}^{0,s-1}(x,y)\right) \oplus \left(y \otimes \widehat{\partial f}_{i,j+1}^{0,s-1}(x,y)\right)$
      end for
    end for
  end for
  $[\widehat{r}_x, \rho_x] = TwoSum(1, -x)$

for $r = 1 : m$ **do**
  for $i = 0$ to $m - r$ **do**
    $[P_{1,x}, \pi_{i0}^{rn}] = TwoProduct(\widehat{r}_x, \widehat{f}_{i0}^{r-1,n}(x,y))$
    $[P_{2,x}, \sigma_{i0}^{rn}] = TwoProduct(x, \widehat{f}_{i+1,0}^{r-1,n}(x,y))$
    $[\widehat{f}_{i0}^{rn}(x,y), \xi_{i0}^{rn}] = TwoSum(P_{1,x}, P_{2,x})$
    $\widehat{l}_{i0}^{rn} = \pi_{i0}^{rn} \oplus \sigma_{i0}^{rn} \oplus \sigma_{i0}^{rn} \oplus \rho_x \otimes \widehat{f}_{i0}^{rn}$
    $\widehat{\partial f}_{i0}^{rn} = \widehat{l}_{i0}^{rn} \oplus \left(\widehat{r}_x \otimes \widehat{\partial f}_{i0}^{r-1,n}\right) \oplus \left(x \otimes \widehat{\partial f}_{i+1,0}^{r-1,n}\right)$
  end for
end for
$res = \widehat{f}_{00}^{mn} \oplus \widehat{\partial f}_{00}^{mn}$

Now an error analysis of the compensated de Casteljau algorithm for the evaluation of tensor product surfaces (Algorithm 6) will be carried out. First, an auxiliary result will be proved.

Lemma 2. *Let* $F(x,y) = \sum_{i=0}^{m}\sum_{j=0}^{n} f_{ij} b_i^m(x) b_j^n(y)$ *be a bivariate polynomial and* $(x,y) \in [0,1] \times [0,1]$. *Then, the de Casteljau algorithm for tensor product surfaces, i.e., Algorithm 5, verifies:*

i. $\sum_{j=0}^{n-s} |f_{ij}^{0s}| b_j^{n-s}(y) \leq \sum_{j=0}^{n-s+1} |f_{ij}^{0,s-1}| b_j^{n-s+1}(y) \leq \cdots \leq \sum_{j=0}^{n} |f_{ij}| b_j^{n-s}(y)$, *for all* $i \in \{0,1,\ldots,m\}$.

ii. $\sum_{i=0}^{m-r} |f_{i0}^{rn}| b_i^{m-r}(x) \leq \sum_{i=0}^{m-r+1} |f_{i0}^{r-1,n}| b_i^{m-r+1}(x) \leq \cdots \leq \sum_{i=0}^{m} |f_{i0}^{0n}| b_i^m(x) \leq \sum_{i=0}^{m}\sum_{j=0}^{n} |f_{ij}| b_i^m(x) b_j^n(y)$.

Proof.

i. Since $f_{ij}^{0s} = (1-y) f_{ij}^{0,s-1} + y f_{i,j+1}^{0,s-1}$ and $y \in [0,1]$ we have

$$|f_{ij}^{0s}| = (1-y)|f_{ij}^{0,s-1}| + y|f_{i,j+1}^{0,s-1}|.$$

By using the recurrence relation of the Bernstein polynomials, $b_i^k(t) = (1-t) b_i^{k-1} + t b_{i-1}^{k-1}(t)$, we can derive

$$\begin{aligned}\sum_{j=0}^{n-s} |f_{ij}^{0s}| b_j^{n-s}(y) &\leq \sum_{j=0}^{n-s} ((1-y)|f_{ij}^{0,s-1}| + y|f_{i,j+1}^{0,s-1}|) b_j^{n-s}(y) \\ &= |f_{i0}^{0,s-1}| b_0^{n-s+1}(y)| + \sum_{j=1}^{n-s} |f_{ij}^{0,s-1}| ((1-y) b_j^{n-j}(y) + u b_{j-1}^{n-j}(y)) + |f_{i,n-s+1}^{0,s-1}| b_{n-s+1}^{n-s+1}(y) \\ &= \sum_{j=0}^{n-s+1} |f_{ij}^{0,s-1}| b_j^{n-s+1}(y).\end{aligned}$$

By iterating this procedure, we obtain all the inequalities in i.

ii. Analogous to i.

□

The error analysis for $\widehat{f}_{00}^{mn}$ was already seen in [7] (and recalled in Theorem 2). Hence, let us see how the roundoff errors affect the computation of $\widehat{\partial f}_{00}^{mn}$ using floating point arithmetic.

Theorem 3. *Let $\widehat{\partial f}_{00}^{mn}$ be the computed value in Algorithm 6 as an approximation of the exact value ∂f_{00}^{mn} in (8). If no underflow occurs, then*

$$|\partial f_{00}^{mn} - \widehat{\partial f}_{00}^{mn}| \leq \gamma_{3(m+n+1)}^2 \sum_{i=0}^{m}\sum_{j=0}^{n} |f_{ij}|\, b_i^m(x) b_j^n(y)$$

Proof. By formula (8) and using i of Lemma 1, we can prove by induction hypothesis on $s \in \{1,\ldots,n\}$ that

$$\widehat{\partial f}_{ij}^{0s} = \partial f_{ij}^{0s} + \sum_{q=1}^{s}\sum_{k=0}^{n-q} l_{i,j+k}^{0q}\theta_{3(n+1-q)+2} b_k^{n-q}(y),$$

for $i = 0,1,\ldots,m$ and $j = 0,1,\ldots,n-s$. Then, analogously, we can also prove by induction hypothesis on $r \in \{1,\ldots,m\}$ that

$$\begin{array}{rl} \widehat{\partial f}_{i0}^{rn} = \partial f_{i0}^{rn} & + \sum_{s=1}^{n}\sum_{k=0}^{r}\sum_{j=0}^{n-s} l_{i+k,j}^{0s}\theta_{3(n+r+1-s)} b_k^r(x) b_j^{n-s}(y) \\ & + \sum_{q=1}^{r}\sum_{k=0}^{r-q} l_{i+k,0}^{qn}\theta_{3(r+1-q)+2} b_k^{r-q}(x), \end{array} \tag{10}$$

for $i = 0,1\ldots,m-r$.

By formula (10) for $r = m$ we can deduce that

$$\begin{array}{rl} \widehat{\partial f}_{00}^{mn} = \partial f_{00}^{mn} & + \sum_{s=1}^{n}\sum_{i=0}^{m}\sum_{j=0}^{n-s} l_{ij}^{0s}\theta_{3(m+n+1-s)} b_i^m(x) b_j^{n-s}(y) \\ & + \sum_{r=1}^{m}\sum_{i=0}^{m-r} l_{i0}^{rn}\theta_{3(m+1-r)+2} b_i^{m-r}(x). \end{array} \tag{11}$$

By Theorem 1 we can derive

$$\begin{array}{rll} |\rho_y| & \leq u\,|r_y| \quad \text{and} & |\widehat{r}_y| \leq (1+u)|r_y| \\ |\pi_{ij}^{0s}| & \leq u\,|\widehat{r}_y \cdot \widehat{f}_{ij}^{0,s-1}| \leq (u+u^2)\,|\widehat{f}_{ij}^{0,s-1}| \cdot |r_y| & \\ |\sigma_{ij}^{0s}| & \leq u\,|y \cdot \widehat{f}_{i,j+1}^{0,s-1}| & \\ |\xi_{ij}^{0s}| & \leq u\,|\widehat{r}_y \otimes \widehat{f}_{ij}^{0,s-1} \oplus y \otimes \widehat{f}_{i,j+1}^{0,s-1}| = u\,|\widehat{r}_y \times \widehat{f}_{ij}^{0,s-1} + y \times \widehat{f}_{i,j+1}^{0,s-1} - \pi_{ij}^{0s} - \sigma_{ij}^{0s}| & \\ & \leq (u+2u^2+u^3)\,|r_y| \cdot |\widehat{f}_{ij}^{0,s-1}| + (u+u^2)\,|y| \cdot |\widehat{f}_{i,j+1}^{0,s-1}| & \end{array} \tag{12}$$

By formulas in (12) we deduce that

$$\begin{array}{rl} |l_{ij}^{0s}| & \leq |\pi_{ij}^{0s}| + |\sigma_{ij}^{0s}| + |\xi_{ij}^{0s}| + |\widehat{f}_{ij}^{0,s-1}| \cdot |\rho_y| \\ & \leq (3u+3u^2+u^3)|r_y| \cdot |\widehat{f}_{ij}^{0,s-1}| + (2u+u^2)|y| \cdot |\widehat{f}_{i,j+1}^{0,s-1}| \\ & \leq 3u\left(|r_y| \cdot |\widehat{f}_{ij}^{0,s-1}| + |y| \cdot |\widehat{f}_{i,j+1}^{0,s-1}|\right) + \mathcal{O}(u^2). \end{array} \tag{13}$$

Analogously we can deduce that

$$|l_{i0}^{rn}| \leq 3u\left(|r_x| \cdot |\widehat{f}_{i0}^{r-1,n}| + |x| \cdot |\widehat{f}_{i+1,0}^{r-1,n}|\right) + \mathcal{O}(u^2). \tag{14}$$

Taking into account that $|r_y| = r_y$, $|r_x| = r_x$, $|x| = x$ and $|y| = y$ for $x, y \in [0,1]$ and using the well known recurrence relation for Bernstein polynomials $b_i^k(t) = (1-t)b_i^{k-1}(t) + t\,b_{i-1}^{k-1}(t)$, it is derived that

$$\begin{array}{rll} \sum_{j=0}^{n-s}\left(|r_y| \cdot |\widehat{f}_{ij}^{0,s-1}| + |y| \cdot |\widehat{f}_{i,j+1}^{0,s-1}|\right) b_j^{n-s}(y) & = \sum_{j=0}^{n-s+1} |\widehat{f}_{ij}^{0,s-1}| \cdot b_j^{n-s+1}(y) & \text{and} \\ \sum_{j=0}^{m-r}\left(|r_x| \cdot |\widehat{f}_{i0}^{r-1,n}| + |x| \cdot |\widehat{f}_{i+1,0}^{r-1,n}|\right) b_i^{m-r}(x) & = \sum_{i=0}^{m-r+1} |\widehat{f}_{i0}^{r-1,n}| \cdot b_i^{m-r+1}(x). & \end{array} \tag{15}$$

By the error analysis of Theorem 2 performed in [7] we have that

$$\widehat{f}_{ij}^{0s} = f_{ij}^{0s}(1+\theta_{3s}) \quad \text{and} \quad \widehat{f}_{i0}^{rn} = f_{i0}^{rn}(1+\theta_{3(r+n)}),$$

where θ_k is a quantity usual in error analysis satisfying that $|\theta_k| \leq \gamma_k$ (for more details see Section 2.1 and [14]). Then we can obtain

$$\sum_{j=0}^{n-s+1} |\widehat{f}_{ij}^{0,s-1}| \cdot b_j^{n-s+1}(y) = (1+\theta_{3(n-1)}) \sum_{j=0}^{n-s+1} |f_{ij}^{0,s-1}| \cdot b_j^{n-s+1}(y) \quad \text{and}$$
$$\sum_{i=0}^{m-r+1} |\widehat{f}_{i0}^{r-1,n}| \cdot b_i^{m-r+1}(x) = (1+\theta_{3(m+n-1)}) \sum_{i=0}^{m-r+1} |f_{i0}^{r-1,n}| \cdot b_i^{m-r+1}(x).$$

Then, applying Lemma 1 and that $|\theta_k| \leq \gamma_k$, we derive

$$\begin{array}{ll} \sum_{j=0}^{n-s+1} |\widehat{f}_{ij}^{0,s-1}| \cdot b_j^{n-s+1}(y) & \leq (1+\gamma_{3(n-1)}) \sum_{j=0}^{n} |\widehat{f}_{ij}| \cdot b_j^{n}(y) \quad \text{and} \\ \sum_{i=0}^{m-r+1} |\widehat{f}_{i0}^{r-1,n}| \cdot b_i^{m-r+1}(x) & \leq (1+\gamma_{3(m+n-1)}) \sum_{i=0}^{m} \sum_{j=0}^{n} |f_{ij}| \cdot b_i^{m}(x) b_j^{n}(y). \end{array} \tag{16}$$

By (13)–(16) we have

$$\begin{array}{ll} \sum_{s=1}^{n} \sum_{j=0}^{n-s} |l_{ij}^{0s}| b_j^{n-s}(y) & \leq 3nu(1+\gamma_{3(n-1)}) \sum_{j=0}^{n} |f_{ij}|\, b_j^{n}(y), \\ \sum_{r=1}^{m} \sum_{i=0}^{m-r} |l_{i0}^{rn}| b_i^{m-r}(x) & \leq 3mu(1+\gamma_{3(m+n-1)}) \sum_{i=0}^{m} \sum_{j=0}^{n} |f_{ij}| b_i^{m}(x) b_j^{n}(y). \end{array}$$

By (11) and taking into account that $3nu(1+\gamma_{3(n-1)}) \leq \gamma_{3n}$ and that $3mu(1+\gamma_{3(m+n-1)}) \leq \gamma_{3(m+n-1)}$ we conclude

$$\begin{aligned} |\widehat{\partial f}_{00}^{mn} - \partial f_{00}^{mn}| \leq \gamma_{3(m+n)}\gamma_{3n} & \sum_{i=0}^{m} \sum_{j=0}^{n} |f_{ij}| b_i^{m}(x) b_j^{n-s}(y) \\ & + \gamma_{3(m+n+1)}\gamma_{3m+2} \sum_{i=0}^{m} \sum_{j=0}^{n} |f_{ij}| b_i^{m}(x) b_j^{n}(y). \end{aligned}$$

Taking into account that $\gamma_{3(m+n)} \leq \gamma_{3(m+n+1)}$ and that, by v of Lemma 1, $\gamma_{3m+2} + \gamma_{3n} \leq \gamma_{3(m+n+1)}$, the result follows. □

Finally, the following result shows the error analysis of the approximation to a tensor product Bézier surface $F(x,y)$ obtained with the compensated de Casteljau algorithm (Algorithm 6).

Theorem 4. *Let $F(x,y) = \sum_{i=0}^{m} \sum_{j=0}^{n} f_{ij} b_i^m(x) b_j^n(y)$ be a tensor product Bézier polynomial with $f_{ij} \in \mathbb{R}$ and res the approximation of $F(x,y)$ computed by Algorithm 6. Then*

$$|F(x,y) - res| \leq u\,|F(x,y)| + \gamma_{3(m+n)+4}^2 S_{B^{mn}}(F(x,y)).$$

Proof. By Algorithm 6 we have that

$$\begin{aligned} |res - F(x,y)| &= |(\widehat{f}_{00}^{mn} \oplus \widehat{\partial f}_{00}^{mn}) - F(x,y)| \\ &= |(1+\delta)(\widehat{f}_{00}^{mn} + \partial f_{00}^{mn} - \partial f_{00}^{mn} + \widehat{\partial f}_{00}^{mn}) - F(x,y)|. \end{aligned}$$

By (9) and taking into account that $|\delta| \leq u$, we deduce

$$|res - F(x,y)| = |(1+\delta)(F(x,y) - \partial f_{00}^{mn} + \widehat{\partial f}_{00}^{mn}) - F(x,y)| \leq u\,|F(x,y)| + (1+u)|\partial f_{00}^{mn} - \widehat{\partial f}_{00}^{mn}|.$$

Then, by Theorem 3 we have

$$|res - F(x,y)| \leq u\,|F(x,y)| + (1+u)\gamma_{3(m+n+1)}^2 S_{B^{mn}}(F(x,y)).$$

Since $(1+u)\gamma_{3(m+n+1)} \leq \gamma_{3(m+n)+4}$ we can deduce

$$|res - F(x,y)| \leq u\,|F(x,y)| + \gamma^2_{3(m+n)+4} S_{B^{mn}}(F(x,y))$$

and the result follows. □

Remark 1. *Assuming that* $(3(m+n)+4)u < 1$, *the error bound for the evaluation of tensor product surfaces by the compensated de Casteljau algorithm obtained in the previous theorem is much lower than the error bound corresponding to the usual de Casteljau algorithm in Theorem 2. The assumption* $(3(m+n)+4)u < 1$ *is typical when working in a CAGD framework. In fact, if*

$$\gamma^2_{3(m+n)+4} \frac{S_{B^{mn}}(F(x,y))}{|F(x,y)|} < u$$

the relative error for the approximation provided by the compensated de Casteljau is u.

4. Conclusions

A compensated version of the de Casteljau algorithm for tensor product functions has been presented. This new method is carried out with the usual floating point arithmetic and operations, and it uses only the same working precision as the data. With this framework, the following bound for the relative error of the new compensated method has been provided:

$$u + \gamma^2_{3(m+n)+4} \frac{S_{B^{mn}}(F(x,y))}{|F(x,y)|},$$

which is lower than the bound corresponding to the usual method

$$\gamma_{3(m+n)} \frac{S_{B^{mn}}(F(x,y))}{|F(x,y)|}.$$

Hence, the new compensated de Casteljau algorithm for tensor product functions can be quite useful for problems with ill-conditioned situations.

Funding: This work was funded by the Spanish research grant PGC2018-096321-B-I00 (MCIU/AEI), by Gobierno de Aragón (E41-17R) and Feder 2014–2020 "Construyendo Europa desde Aragón".

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CAGD	Computer-aided geometric design
EFT	Error-free transformation

References

1. Wei, F.; Zhou, F.; Feng, J. Survey of real root finding of univariate polynomial equation in CAGD/CG. *J. Comput.-Aided Des. Comput. Graph.* **2011**, *23*, 193–207.
2. McNamee, J.M. *Numerical Methods for Roots of Polynomials: Part 1: Volume 14, (Studies in Computational Mathematics)*; Elsevier: Amsterdam, The Netherlands, 2007.
3. Delgado, J.; Peña, J.M. Running Relative Error for the Evaluation of Polynomials. *SIAM J. Sci. Comput.* **2009**, *31*, 3905–3921. [CrossRef]
4. Farouki, R.T.; Rajan, V.T. On the numerical condition of polynomials in Bernstein form. *Comput. Aided Geom. Des.* **1987**, *4*, 191–216. [CrossRef]

5. Farouki, R.T.; Goodman, T.N.T. On the optimal stability of the Bernstein basis. *Math. Comp.* **1996**, *65*, 1553–1566. [CrossRef]
6. Mainar, E.; Peña, J.M. Error analysis of corner cutting algorithms. *Numer. Algorithms* **1999**, *22*, 41–52. [CrossRef]
7. Delgado, J.; Peña, J.M. Error analysis of efficient evaluation algorithms for tensor product surfaces. *J. Comput. Appl. Math.* **2008**, *219*, 156–169. [CrossRef]
8. Delgado, J.; Peña, J.M. Algorithm 960: POLYNOMIAL: An Object-Oriented Matlab Library of Fast and Efficient Algorithms for Polynomials. *ACM Trans. Math. Softw.* **2016**, *42*, 19. [CrossRef]
9. Ogita, T.; Rump, S.M.; Oishi, S. Accurate sum and dot product. *SIAM J. Sci. Comput.* **2005**, *26*, 1955–1988. [CrossRef]
10. Rump, S.M.; Ogita, T.; Oishi, S. Accurate Floating-Point Summation Part I: Faithful Rounding. *SIAM J. Sci. Comput.* **2008**, *31*, 189–224. [CrossRef]
11. Rump, S.M.; Ogita, T.; Oishi, S. Accurate floating-point summation part II: Sign, K-fold faithful and rounding to nearest. *SIAM J. Sci. Comput.* **2008**, *31*, 1269–1302. [CrossRef]
12. Graillat, S.; Langlois, P.; Louvet, N. Algorithms for accurate, validated and fast polynomial evaluation. *Jpn. J. Ind. Appl. Math.* **2009**, *26*, 191–214. [CrossRef]
13. Jiang, H.; Li, S.; Cheng, L.; Su, F. Accurate evaluation of a polynomial and its derivative in Bernstein form. *Comput. Math. Appl.* **2010**, *60*, 744–755. [CrossRef]
14. Higham, N.J. *Accuracy and Stability of Numerical Algorithms*, 2nd ed.; SIAM: Philadelphia, PA, USA, 2002.
15. Lyche, T.; Peña, J.M. Optimally stable multivariate bases. *Adv. Comput. Math.* **2004**, *20*, 149–159. [CrossRef]
16. Peña, J.M. On the optimal stability of bases of univariate functions. *Numer. Math.* **2002**, *91*, 305–318. [CrossRef]
17. Peña, J.M. A note on the optimal stability of bases of univariate functions. *Numer. Math.* **2006**, *103*, 151–154. [CrossRef]
18. Knuth, D.E. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*; Addison Wesley: Boston, MA, USA, 1969.
19. Dekker, T.J. A floating-point technique for extending the available precision. *Numer. Math.* **1971**, *18*, 224–242. [CrossRef]
20. Louvet, N. Algorithmes Compensés en Arithmétique Flottante: Précision, Validation, Performances. Ph.D. Thesis, University of Perpignan, Perpignan, France, 2007.
21. Farin, G. *Curves and Surfaces for Computer Aided Geometric Design*, 5th ed.; Academic Press: San Diego, CA, USA, 2002.
22. Langlois, P.H.; Louvet, N.; Graillat, S. *Compensated Horner Scheme*; Technical Report RR2005-04; Université de Perpignan Via Domitia: Perpignan, France, 2005.
23. Langlois, P.H.; Louvet, N. How to Ensure a Faithful Polynomial Evaluation with the Compensated Horner Algorithm. In Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH'07), Montepellier, France, 25–27 June 2007; pp. 141–149.