INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 18, Number 1, Pages 1–18

EVALUATION OF CURVES AND SURFACES BY METHODS BASED ON DIFFERENTIAL EQUATIONS AND CAGD

JORGE DELGADO AND JUAN MANUEL PEÑA

Abstract. For the evaluation of free-form polynomial and trigonometric curves and surfaces, several Taylor methods and two new methods (DP and DT) motivated by Computer Aided Geometric Design (CAGD) have been considered. Their accuracy and computational costs are compared through numerical examples. In the polynomial case they are also compared with the most used evaluation algorithms.

Key words. Free-form curves and surfaces, evaluation, Taylor method, polynomials, trigonometric polynomials.

1. Introduction

The design of free-form curves and surfaces through a set of control points and a blending system of basis functions is a usual tool in the fields of computer aided design, computer aided manufacturing, geometric modeling or computer graphics. Among the blending bases of univariate functions, we can mention the Bernstein basis of polynomials (see Section 3), the B-splines or the rational Bernstein basis (see [9], [13]). All these bases are examples of normalized B-bases, which are the bases with optimal shape preserving properties of their corresponding spaces (see [2]). Normalized B-bases have also been found for spaces containing other types of functions, such as exponential, trigonometric or hyperbolic functions, which are also useful in many applications (cf. [10], [11], [12], [14], [16], [17], [18], [23], [26]).

The de Casteljau and the rational de Casteljau algorithms are stable algorithms for the evaluation of polynomial or rational Bézier curves and surfaces and present additional advantages over other known evaluation algorithms (cf. [9], [3]). However, for the spaces mentioned at the end of the previous paragraph, algorithms with similar advantages to those of the de Casteljau algorithm have not been found. These and other spaces useful in CAGD (computer aided geometric design) satisfy that their basis functions are the solutions of linear differential systems. In [24] it was shown that typical numerical methods for solving the differential systems (as the Taylor method or the implicit midpoint scheme) can be used to evaluate free-form curves and also to evaluate tensor-product surfaces, which can be generated by evaluating a series of isoparametric curves at a grid of points.

A first goal of our paper is the comparison of the accuracy and computational cost of several dynamic methods related with Taylor method for evaluating algebraic and trigonometric polynomials. In the polynomial case, we consider randomly generated polynomials and Wilkinson's polynomials, and we also compare these methods with the most efficient evaluation methods (see [4], [5]). A second goal of this paper is the presentation of two new evaluation methods for algebraic and trigonometric polynomials, which are called DP (direct polynomial) and DT (direct trigonometric), respectively. Both methods use the normalized B-basis of their corresponding space and perform a direct evaluation, computing the basis functions

Received by the editors March 26, 2019 and, in revised form, October 10, 2020.
2000 *Mathematics Subject Classification.* 65D17, 65D25, 65L05.

in a nested way. The nice behavior of these algorithms is illustrated by numerical examples.

The paper is organized as follows. Section 2 presents several Taylor methods for evaluation. Section 3 presents the evaluation methods for algebraic polynomials and contains the numerical examples comparing them. The corresponding evaluation methods and numerical examples for trigonometric polynomials are presented in Section 4. Finally, Section 5 summarizes the main conclusions of the paper and considers a future work.

2. Evaluations of functions by solving system of linear differential equations

Let us consider representations given by

$$f(t) = \sum_{i=0}^{n} c_{0i}\, u_i(t), \quad t \in [a,b], \tag{1}$$

where $(c_{0i})_{i=0}^{n}$ is a sequence of real numbers and $(u_0, u_1, \ldots, u_n)$ is a basis of a linear space of functions Ω.

If $u_i' \in \Omega$ for all $i = 0, 1, \ldots, n$, then $f(t)$ can be evaluated by solving a system of linear differential equations according to [24]. Let us recall this method. If $u_i' \in \Omega$ for all i then there exists a matrix $R = (r_{ij})_{0\le i,j\le n}$ such that

$$\frac{d}{dt}\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \end{pmatrix} = R \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \end{pmatrix}. \tag{2}$$

Differentiating (1) and taking into account the previous expression we obtain

$$f'(t) = \sum_{i=0}^{n} c_{0i}\, u_i'(t) = (c_{00}, \ldots, c_{0n}) \frac{d}{dt}\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \end{pmatrix} = (c_{00}, \ldots, c_{0n}) R \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \end{pmatrix}.$$

Let us choose $(c_{ij})_{1\le i\le n}^{0\le j\le n}$ such that the matrix $C = (c_{ij})_{0\le i,j\le n}$ is nonsingular. Let us define

$$X(t) := (x_0(t), x_1(t), \ldots, x_n(t))^T = C\,(u_0(t), \ldots, u_n(t))^T, \quad t \in [a,b]. \tag{3}$$

We can observe that $x_0(t) = f(t)$. Differentiating the previous expression and using (2) and (3) we obtain

$$X'(t) := C \frac{d}{dt}\begin{pmatrix} u_0(t) \\ u_1(t) \\ \vdots \\ u_n(t) \end{pmatrix} = C\,R \begin{pmatrix} u_0(t) \\ u_1(t) \\ \vdots \\ u_n(t) \end{pmatrix}.$$

Taking into account that C is nonsingular and denoting $A := CRC^{-1}$ we derive from the previous expression the following system of linear differential equations:

$$X'(t) = AX(t). \tag{4}$$

Let us recall that the first component of the solution of the previous system of differential equations is $x_0(t) = f(t)$. So, each method to solve numerically a system of differential equations can be used as a method to evaluate the function $f(t)$ in (1).

In this paper we consider several Taylor methods for solving those systems.

2.1. Taylor methods. Now let us show how to obtain approximations to $X(t)$ in (3) at a mesh of points $(t_i)_{i=1}^K$ given by $t_i = a + i\,h$, where $h = \frac{b-a}{K}$.

Let us consider the following initial value problem (IVP)

$$\left\{ \begin{array}{ll} X'(t) & = AX(t), \quad t \in [a,b] \\ X(a) & = X_0 \end{array} \right. . \tag{5}$$

The point $X(t_{i+1}) = X(t_i + h)$ is computed by the usual Taylor method as a truncation of the following series

$$\begin{aligned} X(t_{i+1}) = X(t_i + h) &= X(t_i) + h\,X'(t_i) + \frac{h^2}{2!}\,X''(t_i) + \cdots + \frac{h^s}{s!}X^{(s)}(t_i) + \cdots \\ &= X(t_i) + h\,A\,X(t_i) + \frac{h^2\,A^2}{2!}\,X(t_i) + \cdots + \frac{h^s\,A^s}{s!}X(t_i) + \cdots . \end{aligned}$$

For example, we consider the following approximation:

$$X(t_{i+1}) \approx X_{i+1} := \sum_{r=0}^{s} \frac{h^r\,A^r}{r!} X_i.$$

Therefore, the usual Taylor method to solve (5) can be expressed as:

$$X(t_i) \approx X_i = M_h\,X_{i-1}, \quad i = 1, \ldots, K, \quad \text{where } M_h = \sum_{r=0}^{s} \frac{h^r A^r}{r!}, \tag{6}$$

with $A^0 = I$ the identity matrix of order $n+1$. From now on, we will call this method *Taylor 0*. Let us mention that, in [25], a method for computing M_h with no need of Taylor expansions has been presented.

Remark 2.1. Now let us analyze the computational cost of Taylor 0. First M_h must be constructed and this has a computational cost of $\mathcal{O}\left(s(n+1)^3\right)$ elementary operations. Then, (6) must performed, which has a computational cost of $\mathcal{O}(K(n+1)^2)$. Hence, the computational cost of the Taylor 0 method for the evaluation of K points of $X(t)$ is of $\mathcal{O}\left(\max\left\{s(n+1)^3, K(n+1)^2\right\}\right)$ elementary operations.

A point $X(t_i)$ can also be approximated by solving the following IVP

$$\left\{ \begin{array}{ll} X'(t) & = AX(t), \quad t \in [a,b] \\ X(b) & = X_K \end{array} \right. . \tag{7}$$

In this case, approximations to $X(t_i)$ can also be obtained by using the Taylor technique as a truncation of the following series:

$$\begin{aligned} X(t_i) &= X(t_{i+1} - h) \\ &= X(t_{i+1}) - h\,X'(t_{i+1}) + \frac{h^2}{2!}\,X''(t_{i+1}) + \cdots + (-1)^s \frac{h^s}{s!}X^{(s)}(t_{i+1}) + \cdots \\ &= \sum_{r=0}^{\infty} (-1)^r \frac{h^r\,A^r}{r!} X(t_{i+1}). \end{aligned}$$

Therefore, the previous expression gives an alternative Taylor method to approximate the values of a parametric function (1) by solving (7) with the previous formula. This method can be expressed in the following way

$$X(t_i) \approx X_i = N_h\,X_{i+1}, \quad i = K-1, \ldots, 1, 0, \quad \text{where } N_h = \sum_{r=0}^{s} (-1)^r \frac{h^r\,A^r}{r!}. \tag{8}$$

where $N_h = I - h\,A + \frac{h^2\,A^2}{2!} + \cdots + (-1)^s \frac{h^s\,A^s}{s!}$. From now on, we will call this method *Taylor 1*.

Remark 2.2. Analogously to the case of Taylor 0, it can be checked that the computational cost of the Taylor 1 method for the evaluation of K points of $X(t)$ is of $\mathcal{O}\left(\max\left\{s(n+1)^3, K(n+1)^2\right\}\right)$ elementary operations.

In [8] it was considered a two-point Hermite Taylor series expansion given by

$$y(x) = (x-x_1)^s \sum_{i=0}^{s-1} \frac{B_i}{i!}(x-x_2)^i + (x-x_2)^s \sum_{i=0}^{s-1} \frac{A_i}{i!}(x-x_1)^i, \tag{9}$$

where

$$A_i = \frac{d^i}{dx^i}\left[\frac{f(x)}{(x-x_2)^s}\right]_{x=x_1} \quad \text{and} \quad B_i = \frac{d^i}{dx^i}\left[\frac{f(x)}{(x-x_1)^s}\right]_{x=x_2},$$

for $i = 0, 1, \ldots, s-1$. In the following result, expressions for A_i's and B_i's are stated, and a straightforward induction on i can be applied for the proof.

Proposition 2.3. *The coefficients A_i and B_i of the two-point Hermite Taylor series expansion* (9) *are given by*

$$A_i = \frac{1}{(x_1-x_2)^{s+i}} \sum_{k=0}^{i} (-1)^k \binom{i}{k} \binom{s-1+k}{s-1} k!\,(x_1-x_2)^{i-k}\, y^{(i-k)}(x_1),$$

$$B_i = \frac{1}{(x_2-x_1)^{s+i}} \sum_{k=0}^{i} (-1)^k \binom{i}{k} \binom{s-1+k}{s-1} k!\,(x_2-x_1)^{i-k}\, y^{(i-k)}(x_2),$$

for $i = 0, 1, \ldots, s-1$.

The *Taylor two-points* method to solve the following problem

$$\begin{cases} X'(t) = A\,X(t) \\ X(a) = X_0,\ X(b) = X_K \end{cases} \tag{10}$$

can be expressed as follows:

$$X(t_i) = (t_i-a)^s \sum_{r=0}^{s-1} \frac{B_r}{r!}(t_i-b)^r + (t_i-b)^s \sum_{r=0}^{s-1} \frac{A_r}{r!}(t_i-a)^r,$$

where

$$A_r = \frac{1}{(a-b)^{s+r}} \sum_{k=0}^{r} (-1)^k (a-b)^{r-k} \binom{r}{k} \binom{s-1+k}{s-1} k! A^{r-k} X_0,$$

$$B_r = \frac{1}{(b-a)^{s+r}} \sum_{k=0}^{r} (-1)^k (b-a)^{r-k} \binom{r}{k} \binom{s-1+k}{s-1} k! A^{r-k} X_K.$$

Remark 2.4. It can be observed that the problem (10) is overdetermined. But, in the context of CAGD, it is usual that representations (1) satisfy the *endpoint interpolation* property: $f(0) = c_{00}$ and $f(b) = c_{0n}$. Hence, both conditions $X(a) = X_0$ and $X(b) = X_K$ in (10) are a priori known and the solution to the problem (10) is unique. The idea of the Taylor two-points method to solve (10) is taking advantage of those two conditions.

Remark 2.5. It can be checked that the computational cost of the Taylor two-points method for the evaluation of $X(t)$ at K points is of $\mathcal{O}\left(\max\left\{s(n+1)^4, K(n+1)^2\right\}\right)$ elementary operations.

3. Evaluation algorithms for polynomials

3.1. Horner algorithm. Let $\mathcal{P}_n$ be the space of polynomials of degree at most n. Let us consider a polynomial $p \in \mathcal{P}_n$, first represented in the power basis

$$p(t) = \sum_{i=0}^{n} c_i t^i. \tag{11}$$

The most well known algorithm to evaluate the previous polynomial is the Horner algorithm. The pseudocode of this algorithm can be seen in Algorithm 1.

Algorithm 1 *Horner algorithm* for the evaluation of $p \in \mathcal{P}_n$ at t

Require: $t \in [0,1]$ and $(c_i)_{i=0}^n$
Ensure: $res \approx \sum_{i=0}^n c_i t^i$
$res = c_n$
$res = res \cdot t + c_r, \quad r = n-1, \ldots, 0$

The Horner algorithm evaluates a polynomial of degree n with a computational cost of order $\mathcal{O}(n)$.

3.2. De Casteljau algorithm. Let us consider a polynomial $p \in \mathcal{P}_n$ again, but now represented in the Bernstein-Bézier form

$$p(t) = \sum_{i=0}^{n} c_i \, b_i^n(t), \quad 0 \le t \le 1, \tag{12}$$

where $b_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$, $0 \le i \le n$, are the Bernstein polynomials of degree n. This representation is often used in CAGD. In fact, the Bernstein basis is the normalized B-basis of $\mathcal{P}_n$ on $[0,1]$ and so it has optimal shape preserving properties (cf. [1], [2]). A polynomial in the Bernstein-Bézier form can be evaluated by the de Casteljau algorithm, the VS algorithm, the DP algorithm and by the methods in Section 2.

The de Casteljau algorithm is the most usual algorithm for the evaluation of polynomial curves in CAGD (cf. [9]). It also presents advantages for subdivision, degree elevation and degree reduction. The pseudocode corresponding to the de Casteljau algorithm can be seen in Algorithm 2.

Algorithm 2 *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: $res \approx \sum_{i=0}^n c_i b_i^n(t)$
$f_j^0 := c_j, \quad j = 0, \ldots, n$
$f_j^r = (1-t) \cdot f_j^{r-1} + t \cdot f_{j+1}^{r-1}, \quad j = 0, \ldots, n-r, \quad r = 1, \ldots, n$
$res = f_0^n$

The de Casteljau algorithm evaluates a polynomial of degree n with a computational cost of order $\mathcal{O}(n^2)$.

3.3. VS algorithm. The VS algorithm was used in [20] for the evaluation of multivariate polynomials. It is a nested algorithm whose univariate version uses a basis (the VS basis) that coincides, up to scaling, with the Bernstein basis. The VS basis $(z_0^n, z_1^n, \ldots, z_n^n)$ for $\mathcal{P}_n$ is given by

$$z_i^n(t) = t^i(1-t)^{n-i} = \frac{b_i^n(t)}{\binom{n}{i}}.$$

The pseudocode of the VS algorithm can be seen in Algorithm 3. It evaluates a polynomial of $\mathcal{P}_n$ represented with the VS basis with a computational cost of order $\mathcal{O}(n)$.

Algorithm 3 *VS algorithm* for the evaluation of $p \in \mathcal{P}_n$ at t

Require: $t \in [0,1]$ and $(c_i)_{i=0}^n$
Ensure: $res \approx \sum_{i=0}^n c_i z_i^n(t)$
  if $t \geq \frac{1}{2}$ **then**
    $coc = \frac{1-t}{t}$
    $p_0(t) = c_0$
    $p_i(t) = coc\, p_{i-1}(t) + c_i$ for $i = 1, \ldots, n$
    $res = p_n(t)\, t^n$
  else
    $coc = \frac{t}{1-t}$
    $p_n(t) = c_n$
    $p_{n-i}(t) = coc\, p_{n+1-i}(t) + c_{n-i}$ for $i = 1, \ldots, n$
    $res = p_0(t)\, (1-t)^n$.
  end if

3.4. Evaluation of polynomials in Bernstein form by DP. It can be shown that

$$b_{i+1}^n(t) = \frac{t}{1-t} \cdot \frac{n-i}{i+1} \cdot b_i^n(t), \quad i = 0, 1, \ldots, n-1, \quad t \in [0,1).$$

Taking into account the previous relation and that $p(1) = \sum_{i=0}^n c_i b_i^n(1) = c_n$ we can derive Algorithm 4 for the evaluation of a polynomial of degree n (12), which we call direct polynomial (DP) evaluation and uses the Bernstein-Bézier representation. Algorithm 4 evaluates (12) with a computational cost of order $\mathcal{O}(n)$. So, we

Algorithm 4 *Direct polynomial (DP) algorithm* for the evaluation of $p \in \mathcal{P}_n$ at t

Require: $t \in [0,1]$ and $(c_i)_{i=0}^n$
Ensure: $res \approx \sum_{i=0}^n c_i b_i^n(t)$
  if $t < 1$ **then**
    $b = (1-t)^n$
    $res = c_0\, b$
    for $i = 1 : n$ **do**
      $b = \frac{n-i+1}{i} \frac{t}{1-t} b$
      $res = res + c_i\, b$
    end for
  else
    $res = c_n$
  end if

have again an evaluation algorithm of the Bernstein-Bézier representation of a polynomial but with a lower order of computational complexity than the de Casteljau algorithm. The simple idea of direct evaluation with respect to the normalized B-basis of the space and a nested computation for the evaluation of the corresponding basis functions can be extended to other spaces, as shown in Section 4. In addition, its accuracy can compete and even improve that of the other evaluation algorithms of the same complexity. In fact, the following remark shows that the computation of b in the DP algorithm has high relative accuracy (HRA).

Remark 3.1. Given an algorithm using only additions of numbers of the same sign, multiplications and divisions, and assuming that each initial real datum is known to HRA, then it is well known that the output of the algorithm can be computed to HRA (cf. [7, p. 52]). Moreover, in (well implemented) floating point arithmetic, HRA is also preserved even when we perform true subtractions when the operands are original (and so, exact) data (cf. p. 53 of [7] and [6]). So, the sufficient condition to assure the HRA of an algorithm is called "no inaccurate cancellation" (NIC) and it is satisfied if it only uses additions of numbers of the same sign, multiplications, divisions and subtractions (additions of numbers of different sign) of the initial data. Clearly, the computation of b in the DP algorithm is NIC and so it has HRA.

3.5. Evaluation of polynomials by Taylor methods. Let us see how to evaluate a polynomial in the Bernstein-Bézier form or in the power form by using the Taylor methods presented in Section 2.

Let us start with a polynomial represented in the Bernstein-Bézier form. In Section 4 of [24] it was shown that

$$\frac{d}{dt}\begin{pmatrix} b_0^n(t) \\ b_1^n(t) \\ \vdots \\ b_n^n(t) \end{pmatrix} = \underbrace{\begin{pmatrix} -n & -1 & 0 & \cdots & 0 & 0 \\ n & -n+2 & -2 & \cdots & 0 & 0 \\ 0 & n-1 & -n+4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & n-2 & -n \\ 0 & 0 & 0 & \cdots & 1 & n \end{pmatrix}}_{R_1} \begin{pmatrix} b_0^n(t) \\ b_1^n(t) \\ \vdots \\ b_n^n(t) \end{pmatrix}.$$

So, in order to evaluate the polynomial (12) with some of the Taylor mehtods mentioned before first we need to choose a nonsingular matrix $C = (c_{ij})_{0\le i,j\le n}$ such that $c_{0j} = c_j$ for $j = 0, 1, \ldots, n$. Then, taking into account Section 2 we obtain the linear differential equations system $X'(t) = A_1\, X(t)$ with $A_1 = CR_1C^{-1}$.

In order to solve the systems of linear differential equations of Section 2 it is necessary to know the conditions $X(a) = X_0$ and/or $X(b) = X_K$. It is well known that a polynomial $p(t) = \sum_{i=0}^{n} c_i\, b_i^n(t)$ satisfies the endpoint interpolation property: $p(0) = c_0$ and $p(1) = c_n$ (see Remark 2.4). Hence, the initial conditions of the systems are known and given by

$$X(0) = C_0 := (c_{00}, c_{10}, \ldots, c_{n0})^T, \quad X(1) = C_n := (c_{0n}, c_{1n}, \ldots, c_{nn})^T.$$

First, Taylor 0 method can be used for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(t) = A_1\, X(t), & t \in [0,1] \\ X(0) = C_0 \end{cases}. \tag{13}$$

This method for the evaluation will be called *Taylor 0 Bernstein (0 B)*.

Then, Taylor 1 method can be used for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(t) = A_1\, X(t), \quad t \in [0,1] \\ X(1) = C_n \end{cases}. \tag{14}$$

This method for the evaluation will be called *Taylor 1 Bernstein (1 B)*.

Finally, Taylor two-points method can be used for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(t) = A_1\, X(t), \quad t \in [0,1] \\ X(0) = C_0 \text{ and } X(1) = C_n \end{cases}. \tag{15}$$

This method for the evaluation will be called *Taylor two-points Bernstein (2-p B)*. Let us recall that in the field of CAGD the two conditions of (15) are known because of the endpoint interpolation property (see Remark 2.4).

Now, let us see how to evaluate a polynomial (11) represented in the power basis by using the Taylor methods presented in Section 2. It can be seen that

$$\frac{d}{dt}\begin{pmatrix} 1 \\ t \\ t^2 \\ \vdots \\ t^n \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 2 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & n-1 & 0 & 0 \\ 0 & \cdots & \cdots & 0 & n & 0 \end{pmatrix}}_{R_2} \begin{pmatrix} 1 \\ t \\ t^2 \\ \vdots \\ t^n \end{pmatrix}.$$

Now, analogously to the case of the Bernstein-Bézier representation, we need to choose a nonsingular matrix $C = (c_{ij})_{0 \le i,j \le n}$ such that $c_{0j} = c_j$ for $j = 0, 1, \ldots, n$. Then, taking into account Section 2 we obtain the linear differential equations system $X'(t) = A_2\, X(t)$ with $A_2 = C R_2 C^{-1}$. In order to solve the system of linear differential equations of Section 2 it is necessary to know the conditions $X(a) = X_0$ and $X(b) = X_K$. In the case of the power basis the initial conditions are

$$X(0) = C_0, \quad X(1) = S_n := \sum_{j=0}^{n} (c_{0j}, c_{1j}, \ldots, c_{nj})^T.$$

Taylor 0 method can be used for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(t) = A_2\, X(t), \quad t \in [0,1] \\ X(0) = C_0 \end{cases}. \tag{16}$$

This method for the evaluation will be called *Taylor* 0 *Power (0 P)*.

Then, Taylor 1 method can be used for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(t) = A_2\, X(t), \quad t \in [0,1] \\ X(1) = S_n \end{cases}. \tag{17}$$

This method for the evaluation will be called *Taylor* 1 *Power (1 P)*.

Finally, Taylor two-points method can be used for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(t) = A_1\, X(t), \quad t \in [0,1] \\ X(0) = C_0 \text{ and } X(1) = S_n \end{cases}. \tag{18}$$

This method for the evaluation will be called *Taylor two-points Power (2-p P)*. The two conditions of the previous problem are again known.

Remark 3.2. The Taylor methods can be used for obtaining the exact solution of the systems of differential linear equations (13), (14), (15), (16), (17) and (18). For example, in the case of the Taylor 0 B method, when the solution $X(t)$ is a polynomial curve of degree n, the exact solution (up to roundoff errors) is obtained by taking $M_h = \sum_{r=0}^{n} \frac{h^r A_1^r}{r!}$.

3.6. Numerical tests. We have generated randomly 20 polynomials of degree 20, 20 polynomials of degree 60 and 10 polynomials of degree 100, using the power representation. In particular, for each polynomial we have generated randomly its coefficients with respect to the power basis as integers in the interval $[-100, 100]$. The coefficients of the polynomials with respect to the Bernstein basis have been computed by using Algorithm 2 in [5] with exact arithmetic. Then we have evaluated all the polynomials at the points of the mesh $(i/200)_{i=0}^{200}$ with exact arithmetic. In addition, we have also evaluated twenty times the polynomials of degrees 20 and 60, and ten times the polynomials of degree 100, at the same points by the Horner algorithm, de Casteljau, VS and DP algorithms, by solving (13) by Taylor 0 B, by solving (14) by Taylor 1 B, by solving (15) by Taylor 2-p B, by solving (16) by Taylor 0 P, by solving (17) by Taylor 1 P and by solving (18) by Taylor 2-p P. In Taylor 0 (6) and Taylor 1 (8) we have taken s equal to the degree of the polynomial being evaluated, that is, $s = 20$ for the polynomials of degree 20, $s = 60$ for the polynomials of degree 60 and $s = 100$ for the polynomials of degree 100, whereas in Taylor 2-p we have taken s equal to the half of the degree of the evaluated polynomial. Then we have computed the average relative errors at each point of the mesh among all the polynomials of each degree 20, 60 and 100. Then, average relative error and maximum relative error among all points of the mesh have been computed. We have also measured the average computational time used by each algorithm for the evaluation of a polynomial of each degree at the complete mesh of points.

Data for the polynomials of degree 20 can be seen in Figure 1 and Table 1. In particular, in the graphic to the left of Figure 1, relative errors for algorithms using the Bernstein representation can be seen. On the other hand, in the graphic to the right of that figure, relative errors for algorithms using the power representation can be seen. In addition, in Table 1 the average relative error, the maximum relative error and the computational time for each algorithm can be seen. For these polynomials, Horner and DP algorithms present the best behaviour with respect to computational time and maximum relative errors and Horner and de Casteljau are the best with respect to the average relative error. In contrast, Taylor 0 B and 1 P present the worst behaviours with respect to the average and maximal relative errors, although all the algorithms are accurate.

Data for the polynomials of degree 60 can be seen in Figure 2 and Table 2. In particular, in the graphic to the left of Figure 2 relative errors for algorithms using the Bernstein representation can be seen. In the graphic to the right of the figure, relative errors for algorithms using the power representation can be seen. In addition, in Table 2 the average relative error, the maximum relative error and the computational time for each algorithm can be seen. For these polynomials, Horner and DP algorithms again present the lowest computational time. The lowest average and maximum relative errors are provided by Horner and de Casteljau. Again, Taylor 0 B and 1 P present the worst behaviours with respect to the average and maximal relative errors.

TABLE 1. Average and maximum errors, and computational time when evaluating polynomials of degree 20 generated randomly.

Algorithm	Average rel. error	Maximum rel. error	Time
Horner	$2.8914e-16$	$6.2538e-15$	$1.1e-05$
Casteljau	$5.2742e-16$	$6.9918e-15$	$1.4e-04$
DP	$7.8841e-16$	$6.6459e-15$	$6.6e-05$
VS	$1.1255e-14$	$1.3119e-13$	$3.2e-04$
Taylor 0 B	$7.3324e-11$	$2.5276e-09$	$1.6e-03$
Taylor 1 B	$7.5518e-15$	$6.8608e-14$	$1.6e-03$
Taylor 2-p B	$4.9022e-13$	$2.1924e-11$	$1.4e-01$
Taylor 0 P	$5.2198e-15$	$4.7475e-14$	$1.4e-03$
Taylor 1 P	$2.2816e-11$	$4.3559e-10$	$1.4e-03$
Taylor 2-p P	$7.9949e-13$	$2.7330e-11$	$1.4e-01$

Data for the polynomials of degree 100 can be seen in Figure 3 and Table 3. In particular, in the graphic to the left of Figure 3, relative errors for algorithms using the Bernstein representation can be seen. In the graphic to the right of the figure, relative errors for algorithms using the power representation can be seen. On the other hand, in Table 3 the average relative error, the maximum relative error and the computational time for each algorithm can be seen. For these

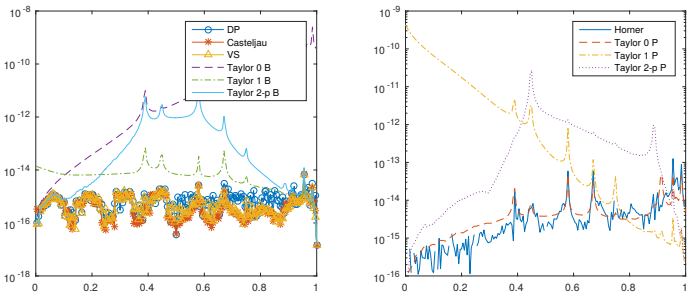

FIGURE 1. Errors when evaluating polynomials of degree 20 generated randomly.

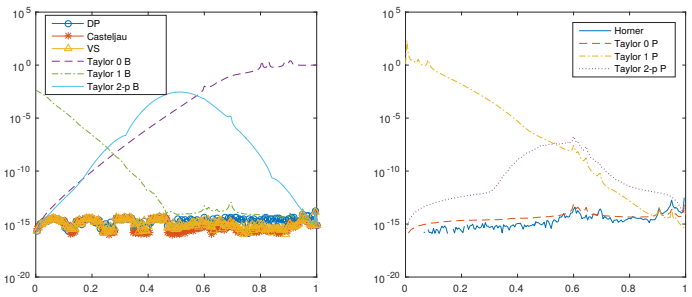

FIGURE 2. Errors when evaluating polynomials of degree 60 generated randomly.

TABLE 2. Average and maximum errors, and computational time when evaluating polynomials of degree 60 generated randomly.

Algorithm	Average rel. error	Maximum rel. error	Time
Horner	$3.4338e-16$	$1.5218e-14$	$2.3e-05$
Casteljau	$1.1973e-15$	$1.5186e-14$	$9.1e-04$
DP	$2.1144e-15$	$1.7051e-14$	$1.2e-04$
VS	$2.8651e-14$	$3.0836e-13$	$8.0e-04$
Taylor 0 B	$2.4505e-01$	$2.6309e+00$	$1.5e-02$
Taylor 1 B	$1.1934e-04$	$4.3496e-03$	$1.4e-02$
Taylor 2-p B	$3.4190e-04$	$2.8575e-03$	$6.4e+00$
Taylor 0 P	$6.4927e-15$	$7.8957e-14$	$1.5e-02$
Taylor 1 P	$1.2537e+00$	$2.0593e+02$	$1.5e-02$
Taylor 2-p P	$7.6784e-09$	$1.7735e-07$	$6.5e+00$

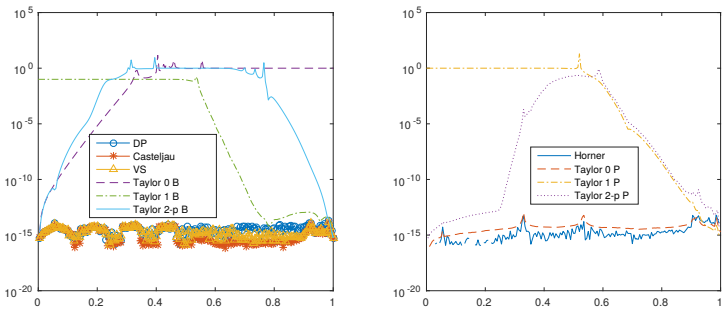

FIGURE 3. Errors when evaluating polynomials of degree 100 generated randomly.

polynomials, Horner and DP algorithms again present the lowest computational time. The lowest average and maximum relative errors are provided by Horner and de Casteljau. Again, Taylor 0 B and 1 P present the worst behaviours with respect to the average and maximal relative errors.

TABLE 3. Average and maximum errors, and computational time when evaluating polynomials of degree 100 generated randomly.

Algorithm	Average rel. error	Maximum rel. error	Time
Horner	$4.1218e-16$	$6.2493e-15$	$5.1e-05$
Casteljau	$1.8522e-15$	$1.4178e-14$	$2.8e-03$
DP	$3.4857e-15$	$2.1790e-14$	$2.2e-04$
VS	$2.3104e-14$	$1.4298e-13$	$1.4e-03$
Taylor 0 B	$7.4928e-01$	$1.5060e+01$	$6.6e-02$
Taylor 1 B	$5.3796e-02$	$1.6599e-01$	$5.9e-02$
Taylor 2-p B	$4.9121e-01$	$8.9741e+00$	$3.9e+01$
Taylor 0 P	$6.8438e-15$	$1.0702e-13$	$6.5e-02$
Taylor 1 P	$6.3560e-01$	$1.9969e+01$	$6.3e-02$
Taylor 2-p P	$3.4907e-02$	$8.3677e-01$	$4.0e+01$

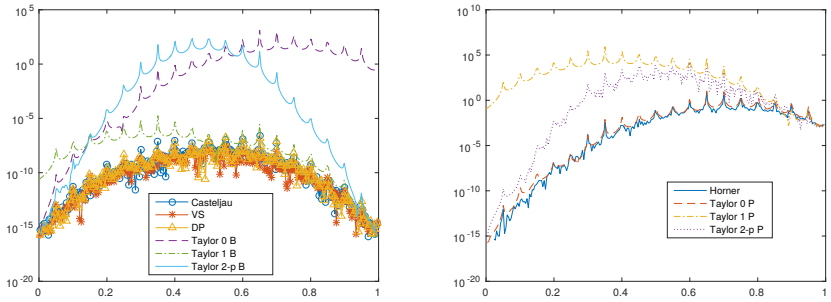

FIGURE 4. Relative errors when evaluating polynomial $p(x)$.

In order to derive some conclusions about the stability of the evaluation algorithms, we also use classical tests. Let us consider the following polynomials

$$p(x) = \prod_{i=1}^{20} \left(x - \frac{i}{20} \right) \quad \text{and} \quad q(x) = \prod_{i=1}^{20} \left(x - \frac{2}{2^i} \right).$$

These polynomials, with roots $\{i/20 \,|\, i = 1, \ldots, 20\}$ and $\{1/2^{i-1} \,|\, i = 1, \ldots, 20\}$, respectively, were originally studied by Wilkinson in [21] and [22]. In these works Wilkinson showed the ill-conditioning of the roots of these polynomials.

First we have computed with Mathematica in exact arithmetic the coefficients of both Wilkinson polynomials with respect to the Bernstein, the VS and the monomial bases. Then we have evaluated polynomial p at the points of the mesh $\{i/257\}_{i=0}^{257}$ with Horner algorithm, de Casteljau, VS and DP algorithms, by solving (13) by Taylor 0 B, by solving (14) by Taylor 1 B, by solving (15) by Taylor 2-p B, by solving (16) by Taylor 0 P, by solving (17) by Taylor 1 P and by solving (18) by Taylor 2-p P, with $s = 20$ for Taylor 0 (6) and Taylor 1 (8), and $s = 10$ for Taylor 2-p. Relative errors can be seen in Figure 4. In particular, in the graphic to the left of Figure 4, relative errors for algorithms using the Bernstein representation can be seen. In the graphic to the right of the figure, relative errors for algorithms using the power representation can be seen. In addition, in Table 4 the average relative error, the maximum relative error and the computational time for each algorithm can be seen. We can observe that only four algorithms are accurate enough for the evaluation $p(x)$: VS, DP, de Casteljau and Taylor 1 B.

Finally we have evaluated the polynomial q at the points of the mesh $\{i/257\}_{i=0}^{257}$ with the same algortihms. Relative errors can be seen in Figure 5. In the graphic to the left of Figure 5, relative errors for algorithms using the Bernstein representation can be seen. In the graphic to the right of the figure, relative errors for algorithms using the power representation can be seen. In addition, in Table 5 the average relative error, the maximum relative error and the computational time for each algorithm can be seen. We can observe that only five algorithms are accurate enough for the evaluation $q(x)$: de Casteljau, DP, VS, Horner and Taylor 0 P.

Remark 3.3. When evaluating a polynomial curve at a small number of points, Taylor methods are not faster than other known evaluation methods for polynomials such as the de Casteljau algorithm. Nevertheless, in CAGD applications it is very often necessary to evaluate a polynomial curve at a large number of points. In this case, the Taylor method is faster than the Casteljau algorithm as it can be seen in

TABLE 4. Average and maximum errors when evaluating polynomial $p(x)$.

Algorithm	Average rel. error	Maximum rel. error
Horner	$1.022525282150209e-01$	$7.384140909590709e+00$
Casteljau	$6.247787735510910e-09$	$2.739654275408099e-07$
DP	$4.922196705116571e-09$	$1.720640597874556e-07$
VS	$1.311800798206712e-09$	$3.176184625997251e-08$
Taylor 0 B	$2.448835183609981e+01$	$1.416183770954575e+03$
Taylor 1 B	$2.654693119489577e-07$	$1.938195540236600e-05$
Taylor 2-points B	$8.449974381220470e+00$	$2.066023751160633e+02$
Taylor 0 P	$1.887436054608656e-01$	$1.100892595124509e+01$
Taylor 1 P	$1.225179560927253e+04$	$8.553363933485990e+05$
Taylor 2-points P	$6.356777995974766e+02$	$2.471774409929074e+04$

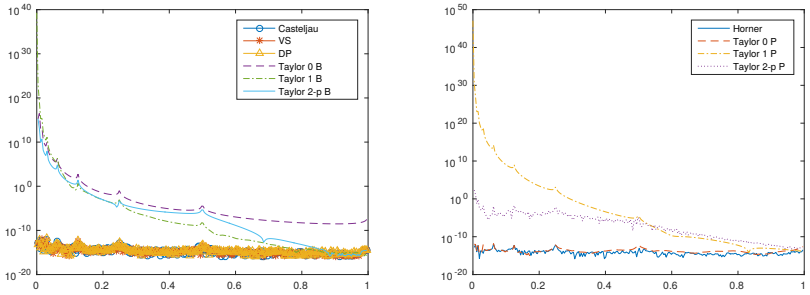

FIGURE 5. Relative errors when evaluating polynomial $q(x)$.

TABLE 5. Average and maximum errors when evaluating polynomial $q(x)$.

Algorithm	Average rel. error	Maximum rel. error
Horner	$2.581593270021876e-14$	$1.080387585722120e-12$
Casteljau	$1.365150114097193e-14$	$6.398300966705923e-13$
DP	$1.644101106265629e-14$	$1.520237677578364e-12$
VS	$1.689130767397024e-14$	$1.520237677578364e-12$
Taylor 0 B	$1.276491769361432e+14$	$3.098650278051026e+16$
Taylor 1 B	$7.579050302369344e+36$	$1.947815927708921e+39$
Taylor 2-points B	$7.002956456764707e+12$	$1.786151714810890e+15$
Taylor 0 P	$5.313254028838989e-14$	$1.542080310162485e-12$
Taylor 1 P	$4.471913215233135e+44$	$1.149281696314916e+47$
Taylor 2-points P	$8.283774684253482e-01$	$1.969320857686634e+02$

the complexity analysis of [24] and the numerical results in Table 2 of Example 1 of that reference.

4. Evaluation of trigonometric polynomials

In [19] it was shown that, for a fixed value of the shape parameter $\beta \in (0, \pi)$, the unique normalized B-basis (that is, the basis with optimal shape preserving

properties) of the vector space of trigonometric polynomials of degree at most n

$$\mathcal{T}_n^\beta = span\{1, \sin u, \cos u, \ldots, \sin(nu), \cos(nu)\}, \quad u \in [0, \beta],$$

is $\{T_{0,2n}^\beta, T_{1,2n}^\beta, \ldots, T_{2n,2n}^\beta\}$, with

$$T_{i,2n}^\beta(u) = t_{i,2n}^\beta \sin^i\left(\frac{u}{2}\right) \sin^{2n-i}\left(\frac{\beta - u}{2}\right), \quad u \in [0, \beta],$$

where $\{t_{i,2n}^\beta\}_{i=0}^{2n}$ are the nonnegative normalizing coefficients defined by

$$t_{i,2n}^\beta = t_{2n-i,2n}^\beta = \frac{1}{\sin^{2n}\left(\frac{\beta}{2}\right)} \sum_{r=0}^{\lfloor \frac{i}{2} \rfloor} \binom{n}{i-r} \binom{i-r}{r} \left(2\cos\left(\frac{\beta}{2}\right)\right)^{i-2r}, \quad i = 0, \ldots, n.$$

As shown in [15], if $\beta \geq \pi$, then there does not exist normalized B-basis of the space $\mathcal{T}_n^\beta$ and, in fact, the space does not possess shape preserving representations, in contrast to our case $\beta < \pi$. For simplicity, from now on we will use t_i instead of $t_{i,2n}^\beta$. Now let us consider a function $f \in \mathcal{T}_{2n}^\beta$ represented by its normalized B-basis as

$$f(u) = \sum_{i=0}^{2n} c_i T_{i,2n}^\beta(u). \tag{19}$$

Analogously to the case of DP algorithm for polynomials in the Bernstein-Bézier form, the previous function can be evaluated by a direct trigonometric (DT) algorithm. For this purpose, it can be checked that

$$T_{i+1,2n}^\beta(u) = \frac{t_{i+1}}{t_i} \cdot \frac{\sin\left(\frac{u}{2}\right)}{\sin\left(\frac{\beta - u}{2}\right)} \cdot T_{i,2n}^\beta(u), \quad i = 0, 1, \ldots, 2n-1, \quad u \in [0, \beta).$$

Taking into account the previous relation and that $f(\beta) = c_{2n}$, we can derive Algorithm 5 for evaluating a trigonometric polynomial (19) by DT algorithm.

Algorithm 5 *Direct trigonometric (DT) algorithm* for the evaluation of $f \in \mathcal{T}_{2n}^\beta$ at u

Require: $\beta \in (0, \pi)$, $u \in [0, \beta]$ and $(c_i)_{i=0}^n$
Ensure: $res \approx \sum_{i=0}^{2n} c_i T_{i,2n}^\beta(u)$
  if $u < \beta$ **then**
    $b = t_0 \sin^{2n}(\frac{\beta - u}{2})$
    $res = c_0\, b$
    for $i = 1 : 2n$ **do**
      $b = \frac{t_i}{t_{i-1}} \frac{\sin\left(\frac{u}{2}\right)}{\sin\left(\frac{\beta - u}{2}\right)} b$
      $res = res + c_i\, b$
    end for
  else
    $res = c_{2n}$
  end if

Analogously to the case of polynomials in Bernstein-Bézier form, in order to evaluate the trigonometric function (19) with some of the Taylor methods mentioned in Section 2, we first need to choose a nonsingular matrix $C = (c_{ij})_{0 \leq i,j \leq 2n}$ such that $c_{0j} = c_j$ for $j = 0, 1, \ldots, 2n$. Then, taking into account Section 2, we obtain

the linear differential equations system $X'(u) = A\,X(u)$ with $A = CRC^{-1}$, where R is given according to Example 2 in [24] by

$$R = \begin{pmatrix} -\frac{n}{\tan\left(\frac{\beta}{2}\right)} & -\frac{t_0}{t_1}\frac{2n}{2\sin\left(\frac{\beta}{2}\right)} & 0 & \cdots & 0 & 0 \\ \frac{t_1}{t_0}\frac{1}{2\sin\left(\frac{\beta}{2}\right)} & -\frac{n-1}{\tan\left(\frac{\beta}{2}\right)} & -\frac{t_1}{t_2}\frac{2n-1}{2\sin\left(\frac{\beta}{2}\right)} & \cdots & 0 & 0 \\ 0 & \frac{t_2}{t_1}\frac{2}{2\sin\left(\frac{\beta}{2}\right)} & -\frac{n-2}{\tan\left(\frac{\beta}{2}\right)} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{n-1}{\tan\left(\frac{\beta}{2}\right)} & -\frac{t_{2n-1}}{t_{2n}}\frac{1}{2\sin\left(\frac{\beta}{2}\right)} \\ 0 & 0 & 0 & \cdots & \frac{t_{2n}}{t_{2n-1}}\frac{2n}{2\sin\left(\frac{\beta}{2}\right)} & \frac{n}{\tan\left(\frac{\beta}{2}\right)} \end{pmatrix}.$$

In order to solve the system of linear differential equations of Section 2 for the particular case of the previous trigonometric polynomials, it is necessary to know the conditions $X(0) = X_0$ and/or $X(\beta) = X_K$. It is straightforward to check that a trigonometric polynomial $f(u) = \sum_{i=0}^{2n} c_i\, T_{i,2n}^{\beta}(u)$ satisfies the endpoint interpolation property: $f(0) = c_0$ and $f(\beta) = c_{2n}$. Hence, the initial conditions of the systems are

$$X(0) = C_0 := (c_{00}, c_{10}, \ldots, c_{2n,0})^T, \quad X(\beta) = C_n := (c_{0,2n}, c_{1,2n}, \ldots, c_{2n,2n})^T.$$

First, let us use Taylor 0 method for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(u) = A\,X(u), & u \in [0, \beta] \\ X(0) = C_0 \end{cases}.$$

Then, Taylor 1 method can be used for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(u) = A\,X(u), & u \in [0, \beta] \\ X(\beta) = C_n \end{cases}.$$

Finally, Taylor two-points method can be used for obtaining an approximation to the solution of the following problem:

$$\begin{cases} X'(u) = A\,X(u), & u \in [0, \beta] \\ X(0) = C_0 \text{ and } X(\beta) = C_n \end{cases}.$$

The two conditions are again known (see Remark 2.4).

4.1. Numerical tests. We have generated randomly 20 trigonometric polynomials of degree 8 represented in the normalized B-basis of $\mathcal{T}_{2n}^{\pi/3}$. In particular, for each function we have generated randomly its 17 coefficients respect to the normalized B-basis $\{T_{0,16}^{\pi/3}, T_{1,16}^{\pi/3}, \ldots, T_{16,16}^{\pi/3}\}$ as integers in the interval $[-100, 100]$. Then we have evaluated all functions at the points of the mesh $(i\pi/600)_{i=0}^{200}$ with exact arithmetic. In addition, we have also evaluated twenty times the functions at the same points by Algortihm 5 and solved the corresponding linear differential system (4) by DT, by Taylor 0 (6) and Taylor 1 (8) both with $s = 33$, and by Taylor two-points with $s = 49$. Then we have computed the average relative errors at each point of the mesh among all functions. Then, the average relative error and the maximum relative error among all points of the mesh are computed. We have also measured the average computational time used by each algorithm for the evaluation of each function at the complete mesh of points. All these data can be seen in Figure 6 and Table 6. We see that the accuracy of DT algorithm is very high, outperforming the remaining algorithms. However, Taylor 1 is also very accurate and presents a similar computational time, and all algorithms are accurate.

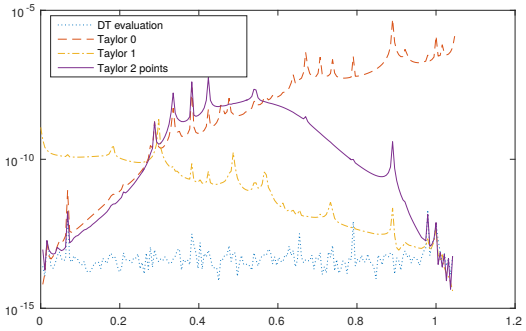

FIGURE 6. Errors when evaluating trigonometric functions generated randomly.

TABLE 6. Average and maximum errors, and computational time when evaluating trigonometric functions generated randomly.

Algorithm	Average rel. error	Maximum rel. error	Time
DT	$3.7005e-15$	$9.8612e-14$	$2.4e-03$
Taylor 0	$5.6254e-09$	$2.5282e-07$	$5.3e-03$
Taylor 1	$3.2336e-12$	$1.1038e-10$	$2.3e-03$
Taylor 2-points	$1.3306e-10$	$2.7572e-09$	$3.2e+00$

In Table 6 it can be seen that the computational time for the evaluation of trigonometric curves at a small number of points is very similar for DT, Taylor 0 and Taylor 1. However, as in the case of the evaluation of polynomials (see Remark 3.3 and the reference therein), when evaluating a trigonometric curve at a large number of points both Taylor 0 and Taylor 1 are faster than other algorithms like DT. In order to illustrate this fact, a trigonometric polynomial of degree 8 has been randomly generated. In particular, we have generated randomly its 17 coefficients respect to the normalized B-basis $\{T^{\pi/3}_{0,16}, T^{\pi/3}_{1,16}, \ldots, T^{\pi/3}_{16,16}\}$ as integers in the interval $[-100, 100]$. Then we have evaluated the function at the points of the mesh $(i\pi/(3\cdot 10^5))_{i=0}^{10^5}$ by DT, by Taylor 0 and Taylor 1 both with $s = 33$, and by Taylor two-points with $s = 49$. The computational time used by each algorithm at the complete mesh of points is shown in Table 7.

TABLE 7. Computational time when evaluating a trigonometric function at a large number of points.

	DT	Taylor 0	Taylor 1	Taylor 2-points
Time	$7.53e-02$	$4.41e-02$	$3.71e-02$	$1.40e+01$

5. Conclusions

Several Taylor methods have been used to evaluate polynomial and trigonometric curves in computer aided design. Direct evaluation methods, called DP and DT for the polynomial and the trigonometric case, respectively, have been presented and they use the normalized B-basis of their corresponding space and a nested

evaluation of their basis functions. Comparisons of these methods and other usual evaluation methods are performed through numerical experiments. It is shown that the accuracy of Taylor methods depends on the place of the evaluation point. The great accuracy of DP and DT is shown. When evaluating at a large number of points, the computational time with Taylor 0 and Taylor 1 is lower than with DP and DT.

Acknowledgments

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

References

[1] Carnicer, J. M. and Peña, J. M., Shape preserving representations and optimality of the Bernstein basis, Adv. Comput. Math., 1 (1993), pp. 173-196.
[2] Carnicer, J. M. and Peña, J. M., Totally positive bases for shape preserving curve design and optimality of B-splines, Comput. Aided Geom. Design, 11 (1994), pp. 633-654.
[3] Delgado, J. and Peña, J. M., A corner cutting algorithm for evaluating rational Bézier surfaces and the optimal stability of the basis, SIAM J. Sci. Comput., 29 (2007), pp. 1668-1682.
[4] Delgado, J. and Peña, J. M., Running Relative Error for the Evaluation of Polynomials, SIAM J. Sci. Comput., 31 (2009), pp. 3905-3921.
[5] Delgado, J. and Peña, J. M., Algorithm 960: POLYNOMIAL: An Object-Oriented Matlab Library of Fast and Efficient Algorithms for Polynomials, ACM Transactions on Mathematical Software, 42 (2016), pp. 23:1-23:19.
[6] Demmel J., Dumitriu I., Holtz O. and Koev P. , Accurate and efficient expression evaluation and linear algebra, Acta Numer., 17 (2008), pp. 87-145.
[7] Demmel, J., Gu, M., Eisenstat, S., Slapnicar, I., Veselic, K. and Drmac, Z., Computing the singular value decomposition with high relative accuracy, Linear Algebra Appl. 299 (1999), pp. 21-80.
[8] Estes, R. H. and Lancaster, E. R., Two-point Taylor Series Expansions, Technical Report NASA-TM-X-55759, 166.
[9] Farin, G., Curves and Surfaces for Computer Aided Geometric Design, 5th ed., Academic Press, San Diego, CA, 2002.
[10] Juhász, I. and Róth, Á., Closed rational trigonometric curves and surfaces, J. Comput. App. Math., 234 (2010), pp. 2390-2404.
[11] Lü, Y., Wang, G. and Yang, X., Uniform hyperbolic polynomial B-spline curves. Comput. Aided Geom. Design, 19 (2002), pp. 379-393.
[12] Goldman, R., The fractal nature of Bézier curves, in Proceedings Geometric Modeling and Processing (GMP 2004), Theory and Applications, Beijing, China, 2004, pp. 3-11.
[13] Goldman, R., An Integrated Introduction to Computer Graphics and Geometric Modeling, CRC Press, Boca Raton, FL, 2009.
[14] Morin, G. and Goldman, R., A subdivision scheme for Poisson curves and surfaces, Comput. Aided Geom. Design, 17 (2000), pp. 813-833.
[15] Peña, J. M., Shape preserving representations for trigonometric polynomial curves, Comput. Aided Geom. Design, 14 (1997), pp. 5-11.
[16] Róth, Á., Control point based exact description of curves and surfaces, in extended Chebyshev spaces, Comput. Aided Geom. Design, 40 (2015), pp. 40-58.
[17] Róth, Á. and Juhász, I., Control point based exact description of a class of closed curves and surfaces, Comput. Aided Geom. Design, 27 (2010), pp. 179-201.
[18] Róth, Á., Juhász, I., Schicho, J. and Hoffmann, M., A cyclic basis for closed curve and surface modeling, Comput. Aided Geom. Design, 26 (2009), pp. 528-46.
[19] Sánchez-Reyes, J., Harmonic rational Bézier curves, p-Bézier curves and trigonometric polynomials, Comput. Aided Geom. Design, 15 (1998), pp. 909-923.
[20] Schumaker, L. L. and Volk, W., Efficient evaluation of multivariate polynomials, Computer Aided Geometric Design, 3 (1986), pp. 149-154.
[21] Wilkinson, J. H., The evaluation of the zeros of ill-conditioned polynomials, Parts I and II, Numer. Math., 1 (1959), pp. 150-180.

[22] Wilkinson, J. H., Rounding Errors in Algebraic Processes, Notes Appl. Sci. 32, Her Majesty's Stationery Office, London, 1963.
[23] Wu, W. and Yang, X., Geometric Hermite interpolation by a family of intrinsically defined planar curves, Comput. Aided Design, 77 (2016), pp. 86-97.
[24] Yang, X. and Hong, J., Dynamic Evaluation of Free-Form Curves and Surfaces, SIAM J. Sci. Comput., 39 (2017), pp. B424-B441.
[25] Yang, X. and Hong, J., Dynamic Evaluation of Exponential Polynomial Curves and Surfaces via Basis Transformation, SIAM J. Sci. Comput., 41 (2019), pp. A3401A3420.
[26] Zhang, J., C-curves: An extension of cubic curves, Comput. Aided Geom. Design, 13 (1996), pp. 199-217.

Departamento de Matemática Aplicada, Universidad de Zaragoza, Teruel, 44003, Spain
E-mail: jorgedel@unizar.es

Departamento de Matemática Aplicada, Universidad de Zaragoza, Zaragoza, 50009, Spain
E-mail: jmpena@unizar.es