Numer Algor (2017) 75:93–111
DOI 10.1007/s11075-016-0198-4

ORIGINAL PAPER

Constrained approximation of rational triangular Bézier surfaces by polynomial triangular Bézier surfaces

Stanisław Lewanowicz[1] · Paweł Keller[2] · Paweł Woźny[1]

Received: 1 December 2015 / Accepted: 22 August 2016 / Published online: 1 September 2016

Abstract We propose a novel approach to the problem of polynomial approximation of rational Bézier triangular patches with prescribed boundary control points. The method is very efficient thanks to using recursive properties of the bivariate dual Bernstein polynomials and applying a smart algorithm for evaluating a collection of two-dimensional integrals. Some illustrative examples are given.

Keywords Rational triangular Bézier surface · Polynomial approximation · Bivariate dual Bernstein basis · Two-dimensional integral · Adaptive quadrature

Mathematics Subject Classification (2010) 41A10 · 65D17 · 65D30 · 33D45

1 Introduction and preliminaries

Rational triangular Bézier surfaces are an important tool in computer graphics. However, they may be sometimes inconvenient in practical applications. The reason is that

✉ Stanisław Lewanowicz
Stanislaw.Lewanowicz@ii.uni.wroc.pl

Paweł Keller
Pawel.Keller@mini.pw.edu.pl

Paweł Woźny
Pawel.Wozny@ii.uni.wroc.pl

[1] Institute of Computer Science, University of Wrocław, ul. F. Joliot-Curie 15, 50-383 Wrocław, Poland

[2] Faculty of Mathematics and Information Science, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warszawa, Poland

evaluation of integrals or derivatives of rational expressions is cumbersome. Also, it happens that a rational surface produced in one CAD system is to be imported into another system which can handle only polynomial surfaces.

In order to solve the two problems above, different algorithms for approximating the rational surface by polynomial surface are proposed [3, 8, 16, 18–20]. The spectrum of methods contains hybrid algorithm [20], progressive iteration approximation [3, 8], least squares approximation and linear programming [16], and approximation by Bézier surfaces with control points obtained by successive degree elevation of the rational Bézier surface [18, 19]. As a rule, no geometric constraints are imposed, which means a serious drawback: if we start with a patchwork of smoothly connected rational Bézier triangles and approximate each patch separately, we do not obtain a smooth composite surface.

In this paper, we propose a method for solving the problem of the constrained least squares approximation of a rational triangular Bézier patch by a polynomial triangular Bézier patch; see Problem 2.1 below. The method is based on the idea of using constrained dual bivariate Bernstein polynomials. Using a fast recursive scheme of evaluation of Bézier form coefficients of the bivariate dual Bernstein polynomials, and applying a swift adaptive scheme of numerical computation of a collection of double integrals involving rational functions contribute to high efficiency of the method.

The outline of the paper is as follows. Section 2 brings a complete solution to the approximation problem. The algorithmic implementation of the method is described in Section 3; some technical details of the implementation are presented in Appendix A. In Section 4, several examples are given to show the efficiency of the method. In Appendix B, some basic information on the Hahn orthogonal polynomials is recalled.

We end this section by introducing some notation. For $\boldsymbol{y} := (y_1, y_2, \ldots, y_d) \in \mathbb{R}^d$, we denote $|\boldsymbol{y}| := y_1 + y_2 + \ldots + y_d$ and $\|\boldsymbol{y}\| := \left(y_1^2 + y_2^2 + \ldots + y_d^2\right)^{\frac{1}{2}}$.

For $n \in \mathbb{N}$ and $\boldsymbol{c} := (c_1,\, c_2,\, c_3) \in \mathbb{N}^3$ such that $|\boldsymbol{c}| < n$, we define the following sets (cf. Fig. 1):

$$\left.\begin{aligned} &\Theta_n := \{\boldsymbol{k} = (k_1, k_2) \in \mathbb{N}^2 :\ 0 \le |\boldsymbol{k}| \le n\},\\ &\Omega_n^{\boldsymbol{c}} := \{\boldsymbol{k} = (k_1, k_2) \in \mathbb{N}^2 :\ k_1 \ge c_1,\ k_2 \ge c_2,\ |\boldsymbol{k}| \le n - c_3\},\\ &\Gamma_n^{\boldsymbol{c}} := \Theta_n \setminus \Omega_n^{\boldsymbol{c}}. \end{aligned}\right\} \tag{1.1}$$

Remark 1.1 The set Θ_n corresponds to the set of control points of a triangular (rational or polynomial) Bézier patch, while the set $\Gamma_n^{\boldsymbol{c}}$ is related with the boundary points, where some constraints are to be imposed. See Section 2 for details.

Throughout this paper, the symbol Π_n^2 denotes the space of all polynomials of two variables, of total degree at most n.

Let T be the standard triangle in $\mathbb{R}^2$,

$$T := \{(x_1,\, x_2) \ :\ x_1,\, x_2 \ge 0,\ x_1 + x_2 \le 1\}. \tag{1.2}$$

For $n \in \mathbb{N}$, and $\boldsymbol{k} := (k_1, k_2) \in \Theta_n$, we denote,

$$\binom{n}{\boldsymbol{k}} := \frac{n!}{k_1!k_2!(n - |\boldsymbol{k}|)!}.$$

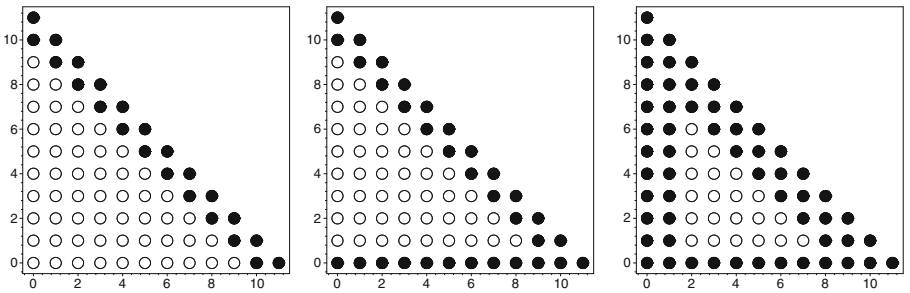

Fig. 1 Examples of sets (1.1) ($n = 11$). Points of the set $\Omega_n^{\boldsymbol{c}}$ are marked by *white discs*, while the points of the set $\Gamma_n^{\boldsymbol{c}}$ – by *black discs*. Obviously, $\Theta_n = \Omega_n^{\boldsymbol{c}} \cup \Gamma_n^{\boldsymbol{c}}$

The *shifted factorial* is defined for any $a \in \mathbb{C}$ by

$$(a)_0 := 1; \qquad (a)_k := a(a+1)\cdots(a+k-1), \qquad k \geq 1.$$

The *Bernstein polynomial basis* in Π_n^2, $n \in \mathbb{N}$, is given by (see, e.g., [5] or [6, §17.3]),

$$B_{\boldsymbol{k}}^n(\boldsymbol{x}) := \binom{n}{\boldsymbol{k}} x_1^{k_1} x_2^{k_2} (1 - |\boldsymbol{x}|)^{n-|\boldsymbol{k}|}, \qquad \boldsymbol{k} := (k_1, k_2) \in \Theta_n, \quad \boldsymbol{x} := (x_1, x_2). \tag{1.3}$$

The (unconstrained) *bivariate dual Bernstein basis polynomials* [12],

$$D_{\boldsymbol{k}}^n(\cdot; \boldsymbol{\alpha}) \in \Pi_n^2, \qquad \boldsymbol{k} \in \Theta_n, \tag{1.4}$$

are defined so that

$$\left\langle D_{\boldsymbol{k}}^n,\ B_{\boldsymbol{l}}^n \right\rangle_{\boldsymbol{\alpha}} = \delta_{\boldsymbol{k},\boldsymbol{l}}, \qquad \boldsymbol{k}, \boldsymbol{l} \in \Theta_n.$$

Here, $\delta_{\boldsymbol{k},\boldsymbol{l}}$ equals 1 if $\boldsymbol{k} = \boldsymbol{l}$, and 0 otherwise, while the inner product is defined by

$$\langle f, g \rangle_{\boldsymbol{\alpha}} := \iint_T w_{\boldsymbol{\alpha}}(\boldsymbol{x})\, f(\boldsymbol{x})\, g(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \tag{1.5}$$

where the weight function $w_{\boldsymbol{\alpha}}$ is given by

$$w_{\boldsymbol{\alpha}}(\boldsymbol{x}) := A_{\boldsymbol{\alpha}} x_1^{\alpha_1} x_2^{\alpha_2} (1 - |\boldsymbol{x}|)^{\alpha_3}, \qquad \boldsymbol{\alpha} := (\alpha_1,\ \alpha_2,\ \alpha_3), \quad \alpha_i > -1, \tag{1.6}$$

with $A_{\boldsymbol{\alpha}} := \Gamma(|\boldsymbol{\alpha}| + 3)/[\Gamma(\alpha_1 + 1)\Gamma(\alpha_2 + 1)\Gamma(\alpha_3 + 1)]$.

Remark 1.2 The normalising constant $A_{\boldsymbol{\alpha}}$ is introduced in order to simplify the form of the dual Bernstein polynomials and some other quantities related to them. See [12] for further explanation.

For $n \in \mathbb{N}$ and $\boldsymbol{c} := (c_1,\ c_2,\ c_3) \in \mathbb{N}^3$ such that $|\boldsymbol{c}| < n$, define the constrained bivariate polynomial space

$$\Pi_n^{\boldsymbol{c},2} := \left\{ P \in \Pi_n^2 \ :\ P(\boldsymbol{x}) = x_1^{c_1} x_2^{c_2} (1 - |\boldsymbol{x}|)^{c_3} \cdot\ Q(\boldsymbol{x}),\ \text{where } Q \in \Pi_{n-|\boldsymbol{c}|}^2 \right\}.$$

It can be easily seen that the constrained set $\{B_{\boldsymbol{k}}^n\}_{\boldsymbol{k} \in \Omega_n^{\boldsymbol{c}}}$ of degree n bivariate Bernstein polynomials forms a basis in this space. We define *constrained dual bivariate*

Bernstein basis polynomials,

$$D_{\boldsymbol{k}}^{(n,\boldsymbol{c})}(\cdot;\boldsymbol{\alpha}) \in \Pi_n^{\boldsymbol{c},2}, \qquad \boldsymbol{k} \in \Omega_n^{\boldsymbol{c}}, \tag{1.7}$$

so that

$$\left\langle D_{\boldsymbol{k}}^{(n,\boldsymbol{c})},\ B_{\boldsymbol{l}}^{n} \right\rangle_{\boldsymbol{\alpha}} = \delta_{\boldsymbol{k},\boldsymbol{l}} \quad \text{for} \quad \boldsymbol{k},\ \boldsymbol{l} \in \Omega_n^{\boldsymbol{c}}, \tag{1.8}$$

where the notation of (1.5) is used. For $\boldsymbol{c} = (0, 0, 0)$, basis (1.7) reduces to the unconstrained basis (1.4) in Π_n^2. Notice that the solution of the least squares approximation problem in the space $\Pi_n^{\boldsymbol{c},2}$ can be given in terms of the polynomials $D_{\boldsymbol{k}}^{(n,\boldsymbol{c})}$. Namely, we have the following result.

Lemma 1.3 *Let F be a function defined on the standard triangle T (cf. (1.2)). The polynomial $S_n \in \Pi_n^{\boldsymbol{c},2}$, which gives the minimum value of the norm*

$$\|F - S_n\|_{L^2,\boldsymbol{\alpha}} := \langle F - S_n,\ F - S_n \rangle_{\boldsymbol{\alpha}}^{\frac{1}{2}},$$

is given by

$$S_n = \sum_{\boldsymbol{k} \in \Omega_n^{\boldsymbol{c}}} \left\langle F,\ D_{\boldsymbol{k}}^{(n,\boldsymbol{c})} \right\rangle_{\boldsymbol{\alpha}} B_{\boldsymbol{k}}^{n}. \tag{1.9}$$

Proof Obviously, S_n has the following representation in the Bernstein basis of the space $\Pi_n^{\boldsymbol{c},2}$:

$$S_n = \sum_{\boldsymbol{k} \in \Omega_n^{\boldsymbol{c}}} \left\langle S_n,\ D_{\boldsymbol{k}}^{(n,\boldsymbol{c})} \right\rangle_{\boldsymbol{\alpha}} B_{\boldsymbol{k}}^{n}.$$

On the other hand, a classical characterization of the best approximation polynomial S_n is that $\langle F - S_n, Q \rangle_{\boldsymbol{\alpha}} = 0$ holds for any polynomial $Q \in \Pi_n^{\boldsymbol{c},2}$ (see, e.g., [4, Thm 4.5.22]). In particular, for $Q = D_{\boldsymbol{k}}^{(n,\boldsymbol{c})}$, we obtain

$$\left\langle F,\ D_{\boldsymbol{k}}^{(n,\boldsymbol{c})} \right\rangle_{\boldsymbol{\alpha}} = \left\langle S_n,\ D_{\boldsymbol{k}}^{(n,\boldsymbol{c})} \right\rangle_{\boldsymbol{\alpha}}, \qquad \boldsymbol{k} \in \Omega_n^{\boldsymbol{c}}.$$

Hence, the formula (1.9) follows. □

The coefficients $E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, n)$ in the Bézier form of the dual Bernstein polynomials,

$$D_{\boldsymbol{k}}^{(n,\boldsymbol{c})}(\boldsymbol{x};\boldsymbol{\alpha}) = \sum_{\boldsymbol{l} \in \Omega_n^{\boldsymbol{c}}} E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, n)\, B_{\boldsymbol{l}}^{n}(\boldsymbol{x}), \qquad \boldsymbol{k} \in \Omega_n^{\boldsymbol{c}}, \tag{1.10}$$

play an important role in the proposed method. Using the duality property (1.8), we obtain the following expression for the coefficients of the above expansion:

$$E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, n) = \left\langle D_{\boldsymbol{k}}^{(n,\boldsymbol{c})},\ D_{\boldsymbol{l}}^{(n,\boldsymbol{c})} \right\rangle_{\boldsymbol{\alpha}}. \tag{1.11}$$

In a recent paper [11], an efficient algorithm was obtained for evaluating all these coefficients for $\boldsymbol{k}, \boldsymbol{l} \in \Omega_n^{\boldsymbol{c}}$, with the computational complexity proportional to the total number of these coefficients. See Section 3.1 for details.

2 Polynomial approximation of Bézier triangular surfaces with constraints

In this paper, we consider the following approximation problem.

Problem 2.1 *Let R_n be a rational triangular Bézier surface of degree n,*

$$\mathsf{R}_n(\boldsymbol{x}) := \frac{\mathsf{Q}_n(\boldsymbol{x})}{\omega(\boldsymbol{x})} = \frac{\sum\limits_{\boldsymbol{k}\in\Theta_n} \omega_{\boldsymbol{k}} r_{\boldsymbol{k}} B_{\boldsymbol{k}}^n(\boldsymbol{x})}{\sum\limits_{\boldsymbol{k}\in\Theta_n} \omega_{\boldsymbol{k}} B_{\boldsymbol{k}}^n(\boldsymbol{x})}, \qquad \boldsymbol{x} \in T,$$

with the control points $r_{\boldsymbol{k}} \in \mathbb{R}^3$ and positive weights $\omega_{\boldsymbol{k}} \in \mathbb{R}$, $\boldsymbol{k} \in \Theta_n$. Find a polynomial Bézier triangular surface

$$\mathsf{P}_m(\boldsymbol{x}) := \sum_{\boldsymbol{k}\in\Theta_m} p_{\boldsymbol{k}} B_{\boldsymbol{k}}^m(\boldsymbol{x}), \qquad \boldsymbol{x} \in T,$$

of degree m, with the control points $p_{\boldsymbol{k}} \in \mathbb{R}^3$, which minimises the distance

$$d_{\boldsymbol{\alpha}}(\mathsf{R}_n, \mathsf{P}_m) := \iint_T w_{\boldsymbol{\alpha}}(\boldsymbol{x}) \|\mathsf{R}_n(\boldsymbol{x}) - \mathsf{P}_m(\boldsymbol{x})\|^2 \,\mathrm{d}\boldsymbol{x}, \tag{2.1}$$

between the surfaces R_n and P_m, and satisfies the additional condition:

$$p_{\boldsymbol{k}} = g_{\boldsymbol{k}} \quad \textit{for} \quad \boldsymbol{k} \in \Gamma_m^{\boldsymbol{c}}, \tag{2.2}$$

where $g_{\boldsymbol{k}} \in \mathbb{R}^3$ are prescribed control points, and $\boldsymbol{c} := (c_1, c_2, c_3) \in \mathbb{N}^3$ is a given parameter vector with $|\boldsymbol{c}| < m$.

Remark 2.2 Remember that continuity conditions for any two adjacent triangular Bézier patches are given in terms of several rows of the control net "parallel" to the control polygon of their common boundary (see, e.g., [6, Section 17.6]). Therefore, constraints (2.2) are natural, in a sense (cf. Fig. 1). In Section 4, we give several examples of practical usage of this approach.

Clearly, the Bézier triangular patch being the solution of Problem 2.1 can be obtained in a componentwise way. Hence, it is sufficient to give a method for solving the above problem in the case where R_n and P_m are scalar functions, and $g_{\boldsymbol{k}}$ are numbers.

All the details of the proposed method are given in the following theorem.

Theorem 2.3 *Assume that we are given the coefficients $r_{\boldsymbol{k}}$ and positive weights $\omega_{\boldsymbol{k}}$, $\boldsymbol{k} \in \Theta_n$, of the rational function*

$$\mathsf{R}_n(\boldsymbol{x}) := \frac{\mathsf{Q}_n(\boldsymbol{x})}{\omega(\boldsymbol{x})} = \frac{\sum\limits_{\boldsymbol{k}\in\Theta_n} \omega_{\boldsymbol{k}} r_{\boldsymbol{k}} B_{\boldsymbol{k}}^n(\boldsymbol{x})}{\sum\limits_{\boldsymbol{k}\in\Theta_n} \omega_{\boldsymbol{k}} B_{\boldsymbol{k}}^n(\boldsymbol{x})}. \tag{2.3}$$

For a fixed $m \in \mathbb{N}$, the polynomial

$$\mathsf{P}_m(\boldsymbol{x}) := \sum_{\boldsymbol{k}\in\Theta_m} p_{\boldsymbol{k}} B_{\boldsymbol{k}}^m(\boldsymbol{x}) \tag{2.4}$$

with the coefficients

$$p_{\boldsymbol{k}} = \sum_{\boldsymbol{l}\in\Omega_m^{\boldsymbol{c}}} \binom{m}{\boldsymbol{l}} E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, m)\big(u_{\boldsymbol{l}} - v_{\boldsymbol{l}}\big), \qquad \boldsymbol{k} \in \Omega_m^{\boldsymbol{c}}, \tag{2.5}$$

where the symbol $E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, m)$ has the meaning given in (1.10),

$$u_{\boldsymbol{l}} := \sum_{\boldsymbol{h}\in\Theta_n} \binom{n}{\boldsymbol{h}} \binom{n+m}{\boldsymbol{h}+\boldsymbol{l}}^{-1} \omega_{\boldsymbol{h}} r_{\boldsymbol{h}}\, I_{\boldsymbol{h}+\boldsymbol{l}},$$

$$v_{\boldsymbol{l}} := \frac{1}{(|\boldsymbol{\alpha}|+3)_{2m}} \sum_{\boldsymbol{h}\in\Gamma_m^{\boldsymbol{c}}} \binom{m}{\boldsymbol{h}} \left(\prod_{i=1}^{3} (\alpha_i+1)_{h_i+l_i}\right) g_{\boldsymbol{h}}$$

with $h_3 := m - |\boldsymbol{h}|$, $l_3 := m - |\boldsymbol{l}|$, and

$$I_{\boldsymbol{j}} := \iint_T w_{\boldsymbol{\alpha}}(\boldsymbol{x}) \frac{B_{\boldsymbol{j}}^{n+m}(\boldsymbol{x})}{\omega(\boldsymbol{x})}\, \mathrm{d}\boldsymbol{x}, \qquad \boldsymbol{j} \in \Omega_{n+m}^{\boldsymbol{c}}, \tag{2.6}$$

minimises the error

$$\|\mathsf{R}_n - \mathsf{P}_m\|_{L^2,\boldsymbol{\alpha}}^2 = \langle \mathsf{R}_n - \mathsf{P}_m, \mathsf{R}_n - \mathsf{P}_m \rangle_{\boldsymbol{\alpha}} \tag{2.7}$$

under the constraints

$$p_{\boldsymbol{k}} = g_{\boldsymbol{k}} \quad \textit{for} \quad \boldsymbol{k} \in \Gamma_m^{\boldsymbol{c}}. \tag{2.8}$$

Proof Observe that

$$\|\mathsf{R}_n - \mathsf{P}_m\|_{L^2,\boldsymbol{\alpha}}^2 = \|\mathsf{W} - \mathsf{S}_m\|_{L^2,\boldsymbol{\alpha}}^2$$

where

$$\mathsf{W} := \mathsf{R}_n - \mathsf{T}_m, \quad \mathsf{T}_m := \sum_{\boldsymbol{k}\in\Gamma_m^{\boldsymbol{c}}} g_{\boldsymbol{k}} B_{\boldsymbol{k}}^m, \quad \mathsf{S}_m := \sum_{\boldsymbol{k}\in\Omega_m^{\boldsymbol{c}}} p_{\boldsymbol{k}} B_{\boldsymbol{k}}^m,$$

the notation being that of (1.1). Thus, we want S_m to be the best approximation polynomial for the function W in the space $\Pi_m^{\boldsymbol{c},2}$. Its Bézier coefficients are given by

$$p_{\boldsymbol{k}} = \left\langle \mathsf{W}, D_{\boldsymbol{k}}^{(m,\boldsymbol{c})} \right\rangle_{\boldsymbol{\alpha}} = \sum_{\boldsymbol{l}\in\Omega_m^{\boldsymbol{c}}} E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, m)\Big(\langle \mathsf{R}_n, B_{\boldsymbol{l}}^m \rangle_{\boldsymbol{\alpha}} - \langle \mathsf{T}_m, B_{\boldsymbol{l}}^m \rangle_{\boldsymbol{\alpha}} \Big), \qquad \boldsymbol{k} \in \Omega_m^{\boldsymbol{c}},$$

where we have used Lemma 1.3. We obtain:

$$\begin{aligned}
\langle \mathsf{R}_n,\, B_{\boldsymbol{l}}^m \rangle_{\boldsymbol{\alpha}} &= \sum_{\boldsymbol{h}\in\Theta_n} \omega_{\boldsymbol{h}} r_{\boldsymbol{h}} \left\langle \frac{B_{\boldsymbol{h}}^n}{\omega},\, B_{\boldsymbol{l}}^m \right\rangle_{\boldsymbol{\alpha}} \\
&= \sum_{\boldsymbol{h}\in\Theta_n} \omega_{\boldsymbol{h}} r_{\boldsymbol{h}} \binom{n}{\boldsymbol{h}} \binom{m}{\boldsymbol{l}} \binom{n+m}{\boldsymbol{h}+\boldsymbol{l}}^{-1} \left\langle \frac{1}{\omega},\, B_{\boldsymbol{h}+\boldsymbol{l}}^{n+m} \right\rangle_{\boldsymbol{\alpha}} \\
&= \sum_{\boldsymbol{h}\in\Theta_n} \omega_{\boldsymbol{h}} r_{\boldsymbol{h}} \binom{n}{\boldsymbol{h}} \binom{m}{\boldsymbol{l}} \binom{n+m}{\boldsymbol{h}+\boldsymbol{l}}^{-1} I_{\boldsymbol{h}+\boldsymbol{l}},
\end{aligned}$$

where we use the notation (2.6). Further, using (1.3) and (1.5), we get

$$\begin{aligned}\langle \mathsf{T}_m, B_{\boldsymbol{l}}^m \rangle_{\boldsymbol{\alpha}} &= \sum_{\boldsymbol{h}\in\Gamma_m^{\boldsymbol{c}}} g_{\boldsymbol{h}} \langle B_{\boldsymbol{h}}^m, B_{\boldsymbol{l}}^m \rangle_{\boldsymbol{\alpha}} \\ &= \sum_{\boldsymbol{h}\in\Gamma_m^{\boldsymbol{c}}} g_{\boldsymbol{h}} \binom{m}{\boldsymbol{h}} \binom{m}{\boldsymbol{l}} \frac{(\alpha_1+1)_{h_1+l_1}(\alpha_2+1)_{h_2+l_2}(\alpha_3+1)_{2m-|\boldsymbol{h}|-|\boldsymbol{l}|}}{(|\boldsymbol{\alpha}|+3)_{2m}}.\end{aligned}$$

Hence, the formula (2.5) follows. □

Remark 2.4 In general, the integrals (2.6) cannot be evaluated exactly. In Section 3.2, we show that they can be efficiently computed numerically up to high precision using an extension of the method of [9] to the case of two-dimensional integration over a triangular domain.

In the special case where all the weights $\omega_{\boldsymbol{i}}$, $\boldsymbol{i} \in \Theta_n$, are equal, the rational function (2.3) reduces to a polynomial of degree n, so that the problem is actually the constrained polynomial degree reduction problem (see, e.g., [17]). Evaluation of the integrals is then a simple task.

3 Implementation of the method

In this section, we discuss the computational aspects of the polynomial approximation of the rational Bézier function described in Section 2 (see Theorem 2.3).

3.1 Computing the coefficients $E_{\boldsymbol{l}}^{\boldsymbol{k}}$

We have to compute all the coefficients $E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, m)$ with $\boldsymbol{k}, \boldsymbol{l} \in \Omega_m^{\boldsymbol{c}}$. It has been shown [11] that they can be given in terms of the quantities

$$e_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\mu}, M) := \langle D_{\boldsymbol{k}}^M, D_{\boldsymbol{l}}^M \rangle_{\boldsymbol{\mu}}, \qquad \boldsymbol{k}, \boldsymbol{l} \in \Theta_M \tag{3.1}$$

with $M := m - |\boldsymbol{c}|$ and $\boldsymbol{\mu} := \boldsymbol{\alpha} + 2\boldsymbol{c}$, where $D_{\boldsymbol{k}}^M(\boldsymbol{x}) \equiv D_{\boldsymbol{k}}^M(\boldsymbol{x}; \boldsymbol{\mu})$ are the unconstrained dual Bernstein polynomials of total degree M (cf. (1.4)). More specifically, we have

$$E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, m) = U\, V_{\boldsymbol{k}}\, V_{\boldsymbol{l}}\, e_{\boldsymbol{l}-\boldsymbol{c}'}^{\boldsymbol{k}-\boldsymbol{c}'}(\boldsymbol{\mu}, M), \qquad \boldsymbol{k}, \boldsymbol{l} \in \Omega_m^{\boldsymbol{c}}, \tag{3.2}$$

where $\boldsymbol{c}' := (c_1, c_2)$, and

$$U := (|\boldsymbol{\alpha}|+3)_{2|\boldsymbol{c}|} \prod_{i=1}^{3} (\alpha_i+1)_{2c_i}^{-1}, \qquad V_{\boldsymbol{h}} := \binom{M}{\boldsymbol{h}-\boldsymbol{c}'} \binom{m}{\boldsymbol{h}}^{-1}.$$

Obviously, we have the following symmetry property: $e_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\mu}, M) = e_{\boldsymbol{k}}^{\boldsymbol{l}}(\boldsymbol{\mu}, M)$ that will be exploited in the Algorithm 3.1 below. This algorithm is based on the following recurrence relations satisfied by $e_{\boldsymbol{l}}^{\boldsymbol{k}} \equiv e_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\mu}, M)$, obtained in [11]. In the sequel, we assume that $e_{\boldsymbol{l}}^{\boldsymbol{k}} = 0$ if $\boldsymbol{k} \notin \Theta_M$ or $\boldsymbol{l} \notin \Theta_M$.

The first recurrence relation is

$$e_{\boldsymbol{l}}^{\boldsymbol{k}+\boldsymbol{v}_2} = \left([\sigma_1(\boldsymbol{k}) - \sigma_1(\boldsymbol{l})] e_{\boldsymbol{l}}^{\boldsymbol{k}} - \sigma_2(\boldsymbol{k}) e_{\boldsymbol{l}}^{\boldsymbol{k}-\boldsymbol{v}_2} + \sigma_0(\boldsymbol{l}) e_{\boldsymbol{l}+\boldsymbol{v}_2}^{\boldsymbol{k}} + \sigma_2(\boldsymbol{l}) e_{\boldsymbol{l}-\boldsymbol{v}_2}^{\boldsymbol{k}} \right) / \sigma_0(\boldsymbol{k}), \tag{3.3}$$

where $\boldsymbol{v}_2 := (0, 1)$, and for $\boldsymbol{t} := (t_1, t_2)$, we define

$$\left.\begin{aligned} \sigma_0(\boldsymbol{t}) := (|\boldsymbol{t}| - M)(t_2 + \mu_2 + 1), \quad \sigma_2(\boldsymbol{t}) := t_2(|\boldsymbol{t}| - \mu_3 - M - 1), \\ \sigma_1(\boldsymbol{t}) := \sigma_0(\boldsymbol{t}) + \sigma_2(\boldsymbol{t}). \end{aligned}\right\} \tag{3.4}$$

To initialise the computation, we use, for $\boldsymbol{l} := (l_1, l_2) \in \Theta_M$, the following formula:

$$e_{\boldsymbol{l}}^{(0,0)}(\mu, M) = \frac{(-1)^{l_1}(|\boldsymbol{\mu}| + 3)_M}{M!(\alpha_1 + 2)_{l_1}} \sum_{i=0}^{M-l_1} C_i(l_1)\, h_i(l_2; \mu_2, \mu_3, M - l_1), \tag{3.5}$$

where

$$C_i(l_1) := \begin{cases} \dfrac{(\mu_1 + 2)_n}{(\mu_2 + \mu_3 + 2)_{M-l_1}}, & i = 0, \\ \dbinom{M}{i} \dfrac{(2i + \mu_2 + \mu_3 + 1)(\mu_1 + 2)_{M-i}(|\boldsymbol{\mu}| + M + 3)_i}{(-M)_i(\mu_3 + 1)_i(\mu_2 + \mu_3 + i + 1)_{M-l_1+1}}, & i \geq 1, \end{cases} \tag{3.6}$$

and we use the notation $h_i(t; a, b, N)$ for the Hahn orthogonal polynomials (see (B.1)). Notice that we can evaluate efficiently the sum in (3.5) using the Clenshaw's algorithm, with the cost of $O(M - l_1)$ operations (see Remark B.1).

The second recurrence relation is

$$e_{\boldsymbol{l}}^{\boldsymbol{k}+\boldsymbol{v}_1} = \left([\tau_1(\boldsymbol{k}) - \tau_1(\boldsymbol{l})]e_{\boldsymbol{l}}^{\boldsymbol{k}} - \tau_2(\boldsymbol{k})\, e_{\boldsymbol{l}}^{\boldsymbol{k}-\boldsymbol{v}_1} + \tau_0(\boldsymbol{l})\, e_{\boldsymbol{l}+\boldsymbol{v}_1}^{\boldsymbol{k}} + \tau_2(\boldsymbol{l})\, e_{\boldsymbol{l}-\boldsymbol{v}_1}^{\boldsymbol{k}}\right)/\tau_0(\boldsymbol{k}), \tag{3.7}$$

where $\boldsymbol{v}_1 := (1, 0)$, and for $\boldsymbol{t} := (t_1, t_2)$, the coefficients $\tau_j(\boldsymbol{t})$ are given by

$$\left.\begin{aligned} \tau_0(\boldsymbol{t}) := (|\boldsymbol{t}| - M)(t_1 + \mu_1 + 1), \quad \tau_2(\boldsymbol{t}) := t_1(|\boldsymbol{t}| - \mu_3 - M - 1), \\ \tau_1(\boldsymbol{t}) := \tau_0(\boldsymbol{t}) + \tau_2(\boldsymbol{t}). \end{aligned}\right\} \tag{3.8}$$

The precise formulation of the algorithm is as follows.

Algorithm 3.1 (Computing the coefficients $E_{\boldsymbol{l}}^{\boldsymbol{k}}(\boldsymbol{\alpha}, \boldsymbol{c}, m)$)

STEP 1 *Let* $M := m - |\boldsymbol{c}|$, *and* $\boldsymbol{\mu} := \boldsymbol{\alpha} + 2\boldsymbol{c}$.

STEP 2 *For* $l_1 = 0, 1, \ldots, M - 1$,
$l_2 = 0, 1, \ldots, M - l_1$,
compute $e_{(l_1,l_2)}^{(0,0)}$ *defined by* (3.5) *and* (3.6) *using the Clenshaw algorithm, and put* $e_{(0,0)}^{(l_1,l_2)} := e_{(l_1,l_2)}^{(0,0)}$.

STEP 3 *For* $k_1 = 0, 1, \ldots, M - 1$,

1^o *for* $k_2 = 0, 1, \ldots, M - k_1 - 1$,
$l_1 = k_1, k_1 + 1, \ldots, M$,
$l_2 = 0, 1, \ldots, M - l_1$,
compute $e_{(l_1,l_2)}^{(k_1,k_2+1)}$ *using the recurrence* (3.3), *and put* $e_{(k_1,k_2+1)}^{(l_1,l_2)} := e_{(l_1,l_2)}^{(k_1,k_2+1)}$;

2^o *for* $l_1 = k_1 + 1, k_1 + 2, \ldots, M$,
$l_2 = 0, 1, \ldots, M - l_1$,
compute $e^{(k_1+1,0)}_{(l_1,l_2)}$ *using the recurrence* (3.7), *and put* $e^{(l_1,l_2)}_{(k_1+1,0)} := e^{(k_1+1,0)}_{(l_1,l_2)}$.

STEP 4 *Compute the table* $\{E^{\boldsymbol{k}}_{\boldsymbol{l}}(\boldsymbol{\alpha}, c, m)\}_{\boldsymbol{k},\boldsymbol{l}\in\Omega^c_m}$ *by* (3.2).

Observe that the complexity order of the algorithm equals $O(m^4)$, i.e., is proportional to the total number of the coefficients $E^{\boldsymbol{k}}_{\boldsymbol{l}}(\boldsymbol{\alpha}, c, m)$.

3.2 Computing the integrals I_j

The most computationally expensive part of the proposed method is the evaluation of the collection of integrals (2.6). For example, for $n + m = 22$, if $\boldsymbol{c} = (0, 0, 0)$, there are 276 two-dimensional integrals to be computed. It is obvious that using any standard quadrature would completely ruin the efficiency of the algorithm. Moreover, if any of the parameters α_i $(i = 1, 2, 3)$ in (1.6) is smaller than 0 and the corresponding constrain parameter c_i equals zero, then the integrands in (2.6) are singular functions, and standard quadratures may fail to deliver any approximations to the integrals.

Therefore, for evaluating the complete set of integrals (2.6), we introduce a special scheme which is based on the general method [9] for approximating singular integrals. The proposed numerical quadrature is of the automatic type, which means that the required number of nodes is adaptively selected, depending on the complexity of the rational Bézier function, so that the requested accuracy of the approximation is always achieved. Most importantly, the algorithm is extremely effective in the considered application. In the example given at the beginning of this subsection $(n + m = 22)$, the time required to compute the whole collection of 276 integrals is only twice[1] longer than the time needed to approximate a single separate integral of a similar type.

First, we shall write the integral (2.6) in a different form which is better suited for fast numerical evaluation. Observe that bivariate Bernstein polynomials (1.3) can be expressed in terms of univariate Bernstein polynomials. Namely, we have

$$B^N_{\boldsymbol{j}}(\boldsymbol{x}) = \mathcal{B}^N_{j_1}(x_1)\mathcal{B}^{N-j_1}_{j_2}(x_2/(1-x_1)), \qquad \boldsymbol{j} := (j_1, j_2),\ \boldsymbol{x} := (x_1, x_2),$$

where $\mathcal{B}^M_i(t) := \binom{M}{i} t^i (1-t)^{M-i}$, $0 \le i \le M$, are univariate Bernstein polynomials of degree M. Further, the bivariate weight function $w_{\boldsymbol{\alpha}}$ (see (1.6)) can be expressed as

$$w_{\boldsymbol{\alpha}}(\boldsymbol{x}) = A_{\boldsymbol{\alpha}}\, v_{\alpha_2+\alpha_3,\alpha_1}(x_1)\, v_{\alpha_3,\alpha_2}(x_2/(1-x_1)),$$

[1] Based on the Maple implementation of the algorithm. If the collection consists of 990 integrals $(n+m = 42)$, the computation time increases by only 50 % (compared to the case of 276 integrals). The detailed report from the efficiency test can be found at the end of Appendix A.

where $v_{\alpha,\beta}(t) := (1-t)^\alpha t^\beta$ is the univariate Jacobi weight function. Hence, the integral (2.6) can be written as

$$\begin{aligned} I_j &= \int_0^1 \int_0^{1-x_1} w_{\boldsymbol{\alpha}}(\boldsymbol{x}) \frac{B_j^N(\boldsymbol{x})}{\omega(\boldsymbol{x})} \,\mathrm{d}x_2 \,\mathrm{d}x_1 \\ &= A_{\boldsymbol{\alpha}} \int_0^1 v_{\alpha_2+\alpha_3+1,\alpha_1}(s) \mathcal{B}_{j_1}^N(s) \left(\int_0^1 v_{\alpha_3,\alpha_2}(t) \frac{\mathcal{B}_{j_2}^{N-j_1}(t)}{\omega^*(s,t)} \,\mathrm{d}t \right) \mathrm{d}s \\ &= A_{\boldsymbol{\alpha}} \binom{N}{\boldsymbol{j}} \int_0^1 v_{a,b}(t) \left(\int_0^1 v_{c,d}(s) \frac{1}{\omega^*(s,t)} \,\mathrm{d}s \right) \mathrm{d}t, \end{aligned} \tag{3.9}$$

where we denoted $N := n + m$,

$$\left.\begin{aligned} a \equiv a(\boldsymbol{j}) &:= \alpha_3 + N - |\boldsymbol{j}|, & b \equiv b(j_2) &:= \alpha_2 + j_2, \\ c \equiv c(j_1) &:= \alpha_2 + \alpha_3 + N - j_1 + 1, & d \equiv d(j_1) &:= \alpha_1 + j_1, \end{aligned}\right\} \tag{3.10}$$

and

$$\omega^*(s,t) := \omega(s,(1-s)t) = \sum_{i=0}^{n} w_i(t) \mathcal{B}_i^n(s), \qquad w_i(t) = \sum_{j=0}^{n-i} \omega_{i,j} \, \mathcal{B}_j^{n-i}(t). \tag{3.11}$$

Note that the computation of values of the integrand is now much more effective, because the coefficients w_i of the function ω^* ($1 \le i \le n$) in (3.11) do not depend of the inner integration variable s. The main idea is, however, to compute the values of ω^* only once (at a properly selected set of quadrature nodes) and obtain a tool for fast computation of the integrals (3.9) for different values of a, b, c, and d, i.e. for different values of $\boldsymbol{j}$.

For arbitrary fixed $t \in [0,1]$, define the function

$$\psi_t(s) := \omega^*(s,t)^{-1}. \tag{3.12}$$

It is easy to see that we can write

$$I_j = A_{\boldsymbol{\alpha}} \binom{N}{\boldsymbol{j}} J(a,b;\Phi),$$

with

$$\Phi(t) := J(c,d;\psi_t), \tag{3.13}$$

where we use the notation

$$J(\alpha,\beta;f) := \int_0^1 (1-x)^\alpha x^\beta f(x) \mathrm{d}x.$$

The functions ψ_t and Φ are analytic in a closed complex region containing the interval $[0,1]$ (it is proved in Appendix A). This implies that (cf. [15, Chapter 3]) they

can be accurately and efficiently approximated by polynomials given in terms of the (shifted) Chebyshev polynomials of the first kind,

$$\begin{aligned} \psi_t(s) &\simeq S_{M_t}(s) := {\sum_{i=0}^{M_t}}' \gamma_i^{[t]} T_i(2s-1), \\ \Phi(t) &\simeq \hat{S}_M(t) := {\sum_{l=0}^{M}}' \hat{\gamma}_l T_i(2t-1), \end{aligned} \qquad 0 \le s,t \le 1, \tag{3.14}$$

where M may depend on j_1, and the prime denotes a sum with the first term halved. Once the above expansions are computed (this can be done in a time proportional to $M_t \log(M_t)$ and $M \log(M)$), the integrals $J(\cdot,\cdot;\cdot)$ can be easily evaluated using the following algorithm that was proved in [14].

Algorithm 3.2 (Computing the integral $J(\alpha,\beta;S)$, S being a polynomial) *Given numbers $\alpha,\ \beta > -1$, let $r := \beta - \alpha$, $u := \alpha + \beta + 1$. Let $S_{\mathcal{M}}$ be a polynomial defined by*

$$S_{\mathcal{M}}(x) = {\sum_{i=0}^{\mathcal{M}}}' \gamma_i T_i(2x-1).$$

Compute the sequence d_i, $0 \le i \le \mathcal{M}+1$, by

$$\begin{aligned} d_{\mathcal{M}+1} &= d_{\mathcal{M}} := 0, \\ d_{i-1} &:= \frac{2rd_i + (i-u)d_{i+1} - \gamma_i}{i+u}, \qquad i = \mathcal{M}, \mathcal{M}-1, \ldots, 1. \end{aligned}$$

Output: $J(\alpha,\beta;S_{\mathcal{M}}) = \mathcal{C}\cdot\left(\frac{1}{2}\gamma_0 - rd_0 + ud_1\right)$, *where* $\mathcal{C} := \Gamma(\alpha+1)\Gamma(\beta+1)/\Gamma(\alpha+\beta+2)$.

By the repeated use of the above very fast scheme, we may efficiently approximate the whole set of integrals $I_{\boldsymbol{j}}$ for $\boldsymbol{j} \in \Omega^c_{n+m}$. The remaining technical details of the adaptive implementation of the proposed quadrature and the complete formulation of the integration algorithm are presented in Appendix A.

3.3 Main algorithm

The method presented in this paper is summarized in the following algorithm.

Algorithm 3.3 (Polynomial approximation of the rational Bézier triangular surface) *Given the coefficients $r_{\boldsymbol{k}}$ and positive weights $\omega_{\boldsymbol{k}}$, $\boldsymbol{k} \in \Theta_n$, of the rational function (2.3), the coefficients $p_{\boldsymbol{k}}$ of the degree m polynomial (2.4) minimising the error (2.7) under the constraints (2.8) can be computed in the following way.*

STEP 1 *Compute the table $\{E_l^{\boldsymbol{k}}(\boldsymbol{\alpha},c,m)\}_{\boldsymbol{k},l\in\Omega^c_m}$ by Algorithm 3.1.*

STEP 2 *Compute the table $\{I_{\boldsymbol{j}}\}_{\boldsymbol{j}\in\Omega^c_{n+m}}$ by Algorithm A.1.*

STEP 3 *For $\boldsymbol{k} \in \Gamma^c_m$, put $p_{\boldsymbol{k}} := g_{\boldsymbol{k}}$.*

STEP 4 *For* $\boldsymbol{k} \in \Omega_m^c$, *compute* $p_{\boldsymbol{k}}$ *by* (2.5).

Output: *Set of the coefficients* $p_{\boldsymbol{k}}$, $\boldsymbol{k} \in \Theta_m$.

4 Examples

In this section, we present some examples of approximation of rational triangular Bézier patches by triangular Bézier patches. No theoretical justification is known for the best choice of the vector parameter $\boldsymbol{\alpha}$ in the distance functional (2.1) if we use the *error function*

$$\Delta_{\boldsymbol{\alpha}}(\boldsymbol{x}) := \|\mathsf{R}_n(\boldsymbol{x}) - \mathsf{P}_m(\boldsymbol{x})\| \tag{4.1}$$

to measure the quality of the approximation. On the base of numerical experiments, we claim that $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$ with $\alpha_i \in (-1, 0)$ $(i = 1, 2, 3)$ usually leads to slightly better results than the one obtained for the more common choice $\boldsymbol{\alpha} = (0, 0, 0)$ (meaning $w_{\boldsymbol{\alpha}}(\boldsymbol{x}) = 1$). The computations were performed in 16-decimal-digit arithmetic. In the implementation of Algorithm A.1, we have assumed $\varepsilon = 5 \times 10^{-16}$ in (A.3) and used the initial values $M^* = M_k^* = 32$.

4.1 Example 1

Let R_6 be the degree 6 rational triangular Bézier patch [8, Example 2],

$$\mathsf{R}_6(\boldsymbol{x}) := \frac{\displaystyle\sum_{\boldsymbol{k}\in\Theta_6} \omega_{\boldsymbol{k}} r_{\boldsymbol{k}} B_{\boldsymbol{k}}^6(\boldsymbol{x})}{\displaystyle\sum_{\boldsymbol{k}\in\Theta_6} \omega_{\boldsymbol{k}} B_{\boldsymbol{k}}^6(\boldsymbol{x})}, \qquad \boldsymbol{x} \in T, \tag{4.2}$$

T being the standard triangle (1.2), and the control points $r_{\boldsymbol{k}}$ and the associated weights $\omega_{\boldsymbol{k}}$ being listed in Table 1. We let $\boldsymbol{\alpha} = (-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$, $\boldsymbol{c} = (1, 1, 1)$ and constructed the degree 5 best approximating polynomial patch

$$\mathsf{P}_5(\boldsymbol{x}) := \sum_{\boldsymbol{k}\in\Theta_5} p_{\boldsymbol{k}} B_{\boldsymbol{k}}^5(\boldsymbol{x}), \qquad \boldsymbol{x} \in T,$$

under the restriction $p_{\boldsymbol{k}} = g_{\boldsymbol{k}}$ for $\boldsymbol{k} \in \Gamma_5^c$, where

$$\Gamma_5^c := \{\boldsymbol{k} = (k_1, k_2) : k_1 = 0, \text{ or } k_2 = 0, \text{ or } |\boldsymbol{k}| = 5\},$$

and the set of points $g_{\boldsymbol{k}}$, $\boldsymbol{k} \in \Gamma_5^c$, is obtained in the following way. As well known, the boundary of the patch (4.2) is formed by three degree 6 rational Bézier curves. The least squares degree 5 polynomial approximation to each of these rational curves, with the endpoints preservation, is constructed using an extension of the method of [14], described in [13] (the input data: $m = 5$, $\alpha = \beta = -\frac{1}{2}$, $k = l = 1$, notation used being that of [13]). Now, the set of points $g_{\boldsymbol{k}}$ is the appropriate collection of all control points of the three resulting Bézier curves. The obtained results are shown in Fig. 2 (upper part).

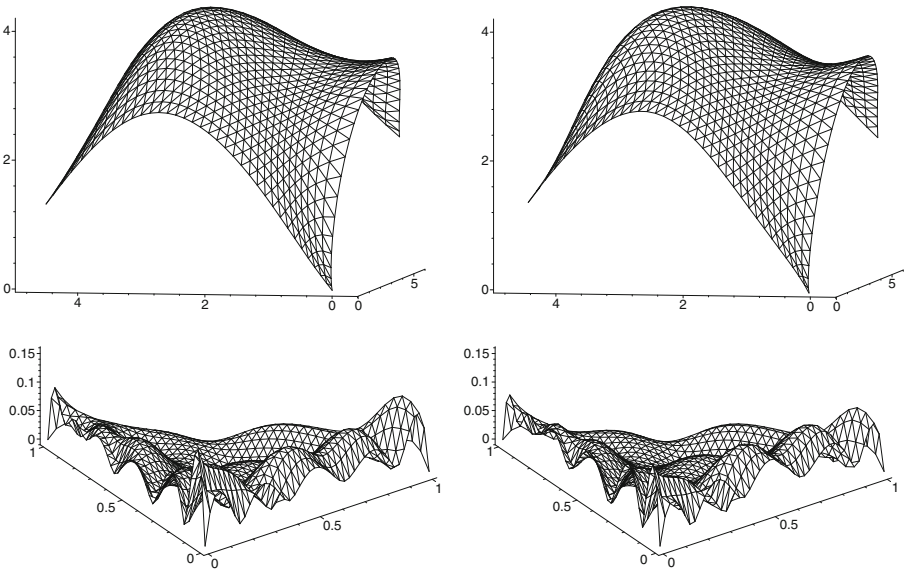

Fig. 2 Constrained degree 5 polynomial approximation of the degree 6 rational triangular Bézier surface, with $\boldsymbol{c} = (1, 1, 1)$. *Upper part*: Rational surface $R_6(\boldsymbol{x})$ and the approximating surface $\mathsf{P}_5(\boldsymbol{x})$ with $\boldsymbol{\alpha} = (-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$. *Lower part*: The error $\Delta_{\boldsymbol{\alpha}}(\boldsymbol{x})$ plots corresponding to $\boldsymbol{\alpha} = (0, 0, 0)$ and $\boldsymbol{\alpha} = (-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$, respectively. Notice that the original surface and the approximating surface agree at the corner points

We have repeated the computations for $\boldsymbol{\alpha} = (a, a, a)$ and a series of values of a (with $\alpha = \beta = a$, in [13]), obtaining the following maximum errors $\mathcal{M}_a :=$

Table 1 Control points $r_{\boldsymbol{k}}$ (*upper entries*) and the associated weights $\omega_{\boldsymbol{k}}$ (*lower entries*) of the surface (4.2), with $\boldsymbol{k} = (k_1, k_2) \in \Theta_6$

$k_1 \setminus k_2$	0	1	2	3	4	5	6
0	(6,0,2)	(5,0,3)	(4,−0.5,3.5)	(3,−0.2,4)	(1.5,0.5,2)	(0.4,0.4,1)	(0,0,0)
	0.8	0.3	1.8	1.2	0.8	0.2	1.6
1	(5.2,1,3)	(4.5,1,3)	(3,0.6,4)	(2,0.9,3)	(1.2,1,2)	(0.4,0.8,0.6)	
	1	0.4	0.8	2.4	1.3	0.9	
2	(4.5,2,5)	(4,2.2,4)	(3,2,3)	(2,1.2,2)	(0.8,1.5,1.5)		
	0.5	1	1	1.8	0.8		
3	(4,3,6)	(2.5,2.5,5)	(1.5,2.8,4)	(1,2,3)			
	0.3	2	1	0.9			
4	(3.5,3.5,4)	(2.5,3,5)	(1.5,3.5,3)				
	1.5	0.6	1.2				
5	(3,4.2,2)	(2,4,2)					
	0.8	0.5					
6	(2,5,1)						

$\max_{\boldsymbol{x}\in T} \Delta_{\boldsymbol{\alpha}}(\boldsymbol{x})$ (cf. (4.1)):

a	$-\frac{7}{8}$	$-\frac{3}{4}$	$-\frac{5}{8}$	$-\frac{1}{2}$	$-\frac{3}{8}$	$-\frac{1}{4}$	$-\frac{1}{8}$	0	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{2}$
$\mathcal{M}_a$	0.122	0.12	0.13	0.132	0.142	0.15	0.157	1.62	0.17	0.176	0.182	0.188

Observe that choosing negative values of a leads to slightly better results than the ones obtained with non-negative a. In Fig. 2 (lower part), we have compared the error $\Delta_{\boldsymbol{\alpha}}(\boldsymbol{x})$ plots corresponding to $a = 0$ and $a = -\frac{1}{2}$. Notice that all the above results are significantly better than the one obtained in [8] by an iterative algorithm with $\boldsymbol{\alpha} = (0, 0, 0)$, where the maximum error was equal to 0.29.

4.2 Example 2

Let R^* be the composite rational surface,

$$\mathsf{R}^*(\boldsymbol{x}) := \begin{cases} \mathsf{R}_5^R(\boldsymbol{y}), & \boldsymbol{y} := (1 - |\boldsymbol{x}|, x_1 - x_2),\ \boldsymbol{x} \in T_R, \\ \mathsf{R}_5^Y(\boldsymbol{z}), & \boldsymbol{z} := (x_2 - x_1, 1 - |\boldsymbol{x}|),\ \boldsymbol{x} \in T_Y, \end{cases} \tag{4.3}$$

where for $C \in \{R, Y\}$,

$$\mathsf{R}_5^C(\boldsymbol{w}) := \frac{\sum_{k\in\Theta_5} \omega_k^C r_k^C B_k^5(\boldsymbol{w})}{\sum_{k\in\Theta_5} \omega_k^C B_k^5(\boldsymbol{w})}, \qquad \boldsymbol{w} \in T, \tag{4.4}$$

T being the standard triangle (1.2), and

$$T_R := \{\boldsymbol{x} = (x_1, x_2) : x_1 \geq x_2 \geq 0,\ |\boldsymbol{x}| \leq 1\},$$

$$T_Y := \{\boldsymbol{x} = (x_1, x_2) : x_2 \geq x_1 \geq 0,\ |\boldsymbol{x}| \leq 1\}.$$

The control points r_k^C and the associated weights ω_k^C of the rational patches (4.4) can be found at the webpage http://www.ii.uni.wroc.pl/~pwo/programs.html. The surface (4.3) is shown in Fig. 3 (the left plot).

Now, we show how to obtain the degree m polynomial approximations of the rational subpatches, which form a C^1-continuous composite surface.

1^o Let P_m^Y be the triangular Bézier patch of degree m approximating the rational patch R_5^Y without constraints, i.e., for $\boldsymbol{c} = (0, 0, 0)$. Let p_k^Y be the control points of the patch P_m^Y.

2^o We approximate the rational patch R_5^R by the triangular Bézier patch P_m^R of degree m, with constraints of the type $\boldsymbol{c} = (2, 0, 0)$, where the points $g_k \in \Gamma_m^{\boldsymbol{c}}$ are chosen so that the C^1-continuity is obtained (cf. [6, Section 17.6]):

$$g_{(0,i)} := p_{(i,0)}^Y, \qquad i = 0, 1, \ldots, m,$$

$$g_{(1,i)} := p_{(i+1,0)}^Y + (p_{(i+1,0)}^Y - p_{(i,1)}^Y), \qquad i = 0, 1, \ldots, m-1.$$

The results, obtained for $m = 5$ and $m = 6$, with $\boldsymbol{\alpha} = (-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$, are shown in Fig. 3. It can be observed that approximation of the rational composite surface (4.3) by two jointed polynomial patches of degree $m = 5$ (the middle plot) resulted in

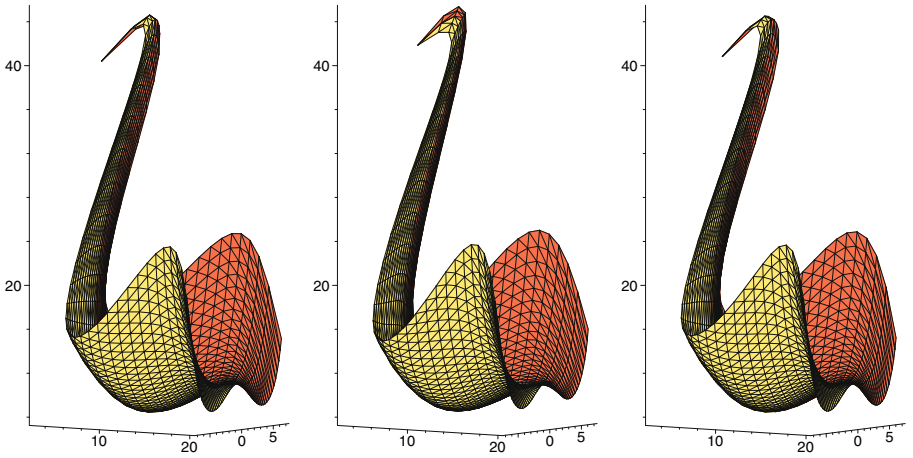

Fig. 3 The composite rational Bézier surface (4.3) (*left*) and the C^1-continuous composite polynomial surfaces of degree (5,5) (*middle*) and (6,6) (*right*)

some visible differences. Increasing the degree of the approximating polynomials to $m = 6$ (the right plot) already gave a very satisfactory result.

5 Conclusions

We propose a method to solve the constrained L^2 approximation of the rational triangular Bézier patch by a polynomial triangular Bézier patch. The algorithm adopts a fast recursive scheme of evaluation of Bézier form coefficients of dual bivariate Bernstein polynomials, and uses an adaptive strategy of numerical computation of a collection of double integrals involving rational functions. The numerical examples confirm high efficiency of the algorithm.

We have shown that the proposed method can be applied in approximation of a C^1-continuous surface composed of two adjacent rational surfaces by a composite polynomial surface, with the preservation of the smoothness order. It would be interesting to extend this approach to the case of a surface composed of more rational surfaces, joined smoothly. However, this may require constructing a separate method for a proper selection of the boundary control points of each part of the approximation surface. Another question worth further research is whether the proposed approach can be modified to the case of G^1-continuity.

Acknowledgments The authors owe thanks to the reviewers for helpful comments and suggestions for improving an earlier version of this manuscript.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

Appendix A: the adaptive algorithm for computing the integrals I_j

We start with proving that the functions ψ_t (3.12), $t \in [0, 1]$, and Φ (3.13) are analytic in a closed complex region containing the interval $[0, 1]$. The assertion is clearly true in the case of $\psi_t(z) = \omega^*(z, t)^{-1}$, as the bivariate polynomial ω^* has no roots in $[0, 1] \times [0, 1]$. Similarly, for any $s \in [0, 1]$, the function $z \mapsto \omega^*(s, z)^{-1}$ is analytic in a rectangular region $[-\sigma, 1 + \sigma] \times [-\sigma, \sigma]$, where $\sigma > 0$ does not depend on s. Thus, if $s \in [0, 1]$, then

$$\int_C \omega^*(s, z)^{-1} \mathrm{d}z = 0$$

for any closed contour $C \subset [-\sigma, 1+\sigma] \times [-\sigma, \sigma]$. Consequently, if $\alpha, \beta > -1$, then

$$\int_C \left(\int_0^1 (1 - s)^\alpha s^\beta \omega^*(s, z)^{-1} \mathrm{d}s \right) \mathrm{d}z = \int_0^1 (1 - s)^\alpha s^\beta \left(\int_C \omega^*(s, z)^{-1} \mathrm{d}z \right) \mathrm{d}s = 0.$$

Therefore, by Morera's theorem (see, e.g., [1, Chapter 2.3]), the function $\Phi(z) = J(\alpha, \beta, \psi_z)$ is also analytic in $[-\sigma, 1 + \sigma] \times [-\sigma, \sigma]$.

The polynomials S_{M_t} and $\hat{S}_M$ in (3.14), which approximate the functions ψ_t and Φ, are determined to satisfy the interpolation conditions

$$\left.\begin{aligned} &S_{M_k}(s_j) = \omega^*(s_j, t_k)^{-1}, \quad 0 \le j \le M_k, \\ &\hat{S}_M(t_k) = J(c, d; S_{M_k}), \end{aligned}\right\} \qquad 0 \le k \le M,$$

where, for simplicity, we denote $M_k \equiv M_{t_k}$, and the interpolation nodes are given by

$$s_j = \frac{1}{2} + \frac{1}{2} \cos \frac{j\pi}{M_k}, \qquad t_k = \frac{1}{2} + \frac{1}{2} \cos \frac{k\pi}{M}. \tag{A.1}$$

In such a case, the coefficients $\gamma_i^{[t_k]}$ and $\hat{\gamma}_l$ in (3.14) are given by

$$\begin{aligned} \gamma_i^{[t_k]} &= \frac{2 - \delta_{i,M_k}}{M_k} \sum_{j=0}^{M_k}{}'' \, \omega^*(s_j, t_k)^{-1} \cos \frac{ij\pi}{M_k}, \qquad 0 \le i \le M_k, \\ \hat{\gamma}_l &= \frac{2 - \delta_{l,M}}{M} \sum_{k=0}^{M}{}'' \, J(c, d; S_{M_k}) \cos \frac{lk\pi}{M}, \qquad 0 \le l \le M, \end{aligned} \tag{A.2}$$

where $\delta_{j,k}$ is the Kronecker delta, the double prime means that the first and the last term of the sum are to be halved. The sets of coefficients (A.2) can be very efficiently computed by means of the FFT with only $O\big(M_k \log(M_k)\big)$ and $O\big(M \log(M)\big)$ arithmetic operations (cf. [7] or [4, Section 5.1]; the authors recall that the FFT is not only fast, but also resistant to round-off errors). The presented approach is very convenient from the practical point of view because if the accuracy of the approximation (3.14) is not satisfactory, then we may double the value of M_k (or M) and reuse the previously computed results. The expansions (3.14) are accepted if

$$\frac{\sum_{i=M_k-3}^{M_k} |\gamma_i^{[t_k]}|}{\max\left\{1, \max_{0 \le i \le 3} |\gamma_i^{[t_k]}|\right\}} \le 16\varepsilon \quad \text{and} \quad \frac{\sum_{i=M-3}^{M} |\hat{\gamma}_i|}{\max\left\{1, \max_{0 \le i \le 3} |\hat{\gamma}_i|\right\}} \le 256\varepsilon, \tag{A.3}$$

where ε is the computation precision.

Here is the complete algorithm for efficient approximation of the whole set of integrals I_j for $\boldsymbol{j} \in \Omega^c_{n+m}$. The functions (parameters) a, b, c, and d are defined in (3.10).

Algorithm A.1 (Numerical computation of the set of integrals I_j, $\boldsymbol{j} \in \Omega^c_{n+m}$)

Let $M := M^*$, where M^* is an arbitrary integer greater than 7.

***Phase I**. For $k \in \{0, 1, \ldots, M\}$, do the following steps 1–6:*

Step 1. *Compute t_k according to (A.1) and compute $w_i(t_k)$ in (3.11) for $i \in \{0, 1, \ldots, n\}$.*

Step 2. *Let $M_k := M_k^*$, where M_k^* is an arbitrary integer greater than 7.*

Step 3. *Compute the values $\omega^*(s_j, t_k)^{-1}$ for $j \in \{0, 1, \ldots, M_k\}$, where s_j is given by (A.1).*

Step 4. *Using the FFT, compute the coefficients $\gamma_i^{[t_k]}$ $(0 \le i \le M_k)$ defined in (A.2).*

Step 5. *If the first condition of (A.3) is not satisfied, then set $M_k := 2M_k$, compute the additional values $\omega^*(s_j, t_k)^{-1}$ for $j \in \{1, 3, 5, \ldots, M_k - 1\}$, and go to step 4.*

Step 6. *Compute the set of quantities $W[t_k, j_1] := J\left(c(j_1), d(j_1); S_{M_k}\right)$ by applying Algorithm 3.2, for $j_1 \in \{c_1, c_1+1, \ldots, N-c_2-c_3\}$, where $N = n + m$.*

***Phase II**. For $j_1 \in \{c_1, c_1 + 1, \ldots, N - c_3 - c_2\}$, perform the following steps 7–9:*

Step 7. *Compute the coefficients $\hat{\gamma}_l$ $(0 \le l \le M)$ defined in (A.2), by means of the FFT, using the stored values $W[t_k, j_1]$, $0 \le k \le M$, in place of $J\left(c(j_1), d(j_1); S_{M_k}\right)$.*

Step 8. *If the second condition of (A.3) is not satisfied, then set $M := 2M$, and repeat Steps 1–6 for $k \in \{1, 3, 5, \ldots, M - 1\}$.*

Step 9. *For $j_2 \in \{c_2, c_2 + 1, \ldots, N - c_3 - j_1\}$, compute the integrals*

$$I_j \equiv I_{(j_1, j_2)} := A_{\boldsymbol{\alpha}} \binom{N}{\boldsymbol{j}} J\left(a(\boldsymbol{j}), b(j_2); \hat{S}_M\right)$$

using Algorithm 3.2.

***Output**: Set of the integrals I_j for $\boldsymbol{j} \in \Omega^c_{n+m}$.*

Remark A.2 In steps 4 and 7 of the above algorithm, the coefficients $\gamma_i^{[t_k]}$ $(0 \le i \le M_k)$ or $\hat{\gamma}_l$ $(0 \le l \le M)$ are recalculated each time the value of M_k or M is doubled. Such a procedure is advised if we use a system (like, e.g., Maple or Matlab) equipped with a fast built-in FFT subroutine. If we are to program the FFT summation algorithm by ourselves, it should rather be done in such a way that practically all results computed for a previous value of M_k or M are reused (cf., e.g., [7]).

In Table 2, we present the results of the efficiency test, where the proposed quadrature (implemented in Maple) is compared to the Maple built-in integration subroutine. We have used the Bézier surface from Example 4.1 ($n = 6$), and set the parameters m and $\boldsymbol{c}$ to several different values, to obtain collections of integrals of different sizes (equal to $|\Omega^{\boldsymbol{c}}_{n+m}|$). The experiment was performed in the 64-bit version of Maple 16 on the computer equipped with the 3.7 GHz i7 processor. All parameters α_i in (1.6) were set to 0 (the efficiency of the proposed method does not depend on $\boldsymbol{\alpha}$, but the Maple built-in integration subroutine works most efficiently with this selection).

We have to keep in mind that Maple is an interpretative programming language with a pretty slow code interpreter. Therefore, the 4.7 times longer computation time of our quadrature, compared to the computation time of the Maple library function, in the case of 1-element collection of integrals is in fact an excellent result. The last collection of 990 integrals ($n + m = 42$) was too difficult to be computed by the Maple built-in subroutine (in 14-decimal digit arithmetic, assumed during this test).

Appendix B: Hahn orthogonal polynomials

The notation

$$ {}_rF_s\left(\left.\begin{matrix} a_1, \ldots, a_r \\ b_1, \ldots, b_s \end{matrix}\right| z\right) := \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_r)_k}{k!(b_1)_k \cdots (b_s)_k}\, z^k $$

is used for the *generalized hypergeometric series* (see, e.g., [2, §2.1]); here, $r, s \in \mathbb{Z}_+$, $z, a_1, \ldots, a_r, b_1, \ldots, b_s \in \mathbb{C}$, and $(c)_k$ is the shifted factorial. The *Hahn polynomials* (see, e.g., [10, §1.5])

$$ h_l(t) \equiv h_l(t; a, b, M) := (a+1)_l (-M)_l \, {}_3F_2\left(\left.\begin{matrix} -l, l+a+b+1, -t \\ a+1, -M \end{matrix}\right| 1\right), \quad l = 0, 1, \ldots, M, \tag{B.1} $$

where $a, b > -1$, and $M \in \mathbb{N}$, satisfy the recurrence relation

$$ h_{l+1}(t) = A_l(t; M)\, h_l(t) + B_l(M)\, h_{l-1}(t), \qquad l \geq 0;\ h_0(t) \equiv 1;\ h_{-1}(t) \equiv 0, \tag{B.2} $$

Table 2 Comparison of the computation times of the Maple library function and the proposed adaptive quadrature (Algorithm A.1) in the case of several collections of integrals (2.6)

$\lvert\Omega^{\boldsymbol{c}}_{n+m}\rvert$	Computation time (in seconds)	
	Maple library function	The proposed method
1	0.064	0.30
3	0.19	0.30
10	0.64	0.32
28	1.75	0.37
91	6.34	0.43
276	22.9	0.59
990	FAILURE	0.89

The number of integrals which are to be computed equals $|\Omega^{\boldsymbol{c}}_{n+m}|$

with the coefficients

$$A_l(t; M) := C_l\,(2l + s - 1)_2\,t - D_l - E_l, \qquad B_l(M) := -D_l\,E_{l-1}, \qquad \text{(B.3)}$$

where $s := a + b + 1$, $C_l := (2l + s + 1)/[(l + s)(2l + s - 1)]$, $D_l := C_l\,l(l + M + s)(l + b)$, and $E_l := (l + a + 1)(M - l)$.

Remark B.1 A linear combination of Hahn polynomials, $s_N(t) := \sum_{i=0}^{N} \gamma_i\,h_i(t; a, b, M)$, can be summed using the following *Clenshaw's algorithm* (see, e.g., [4, Thm 3.2.11]). Compute the sequence $V_0, V_1, \ldots, V_{n+2}$ from

$$V_i := \gamma_i + A_i(t; M)V_{i+1} + B_{i+1}(M)V_{i+2}, \qquad i = N, N-1, \ldots, 0,$$

with $V_{N+1} = V_{N+2} = 0$, where the coefficients $A_i(t; M)$ and $B_i(M)$ are defined by (B.3). Then, $s_N(t) = V_0$.

References

1. Ahlfors, L.V.: Complex Analysis, 3rd edn. McGraw-Hill (1979)
2. Andrews, G.E., Askey, R., Roy, R.: Special Functions, Encyclopedia of Mathematics and its Applications, vol. 71. Cambridge University Press, Cambridge (1999)
3. Chen, J., Wang, G.J.: Progressive-iterative approximation for triangular Bézier surfaces. Comp. Aided-Des. **43**, 889–895 (2011)
4. Dahlquist, G., Björck, A.: Numerical Methods in Scientific Computing, vol. I. Society for Industrial and Applied Mathematics, Philadelphia (2008)
5. Farin, G.: Triangular Bernstein-Bézier patches. Comput. Aided Geom. Des. **3**, 83–127 (1986)
6. Farin, G. Curves and Surfaces for Computer-Aided Geometric Design. A Practical Guide, 5t. Academic Press, Boston (2002)
7. Gentleman, W.M.: Implementing Clenshaw-Curtis quadrature – II. Computing the cosine transformation. Commun. ACM **15**, 343–346 (1972)
8. Hu, Q.Q.: An iterative algorithm for polynomial approximation of rational triangular Bézier surfaces. Appl. Math. Comput. **219**, 9308–9316 (2013)
9. Keller, P.: A method for indefinite integration of oscillatory and singular functions. Numer. Algor. **46**, 219–251 (2007)
10. Koekoek, R., Swarttouw, R.F.: The Askey scheme of hypergeometric orthogonal polynomials and its q-analogue. Fac. Techn. Math. Informatics, Delft Univ. of Technology, Rep. 98-17, Delft (1998)
11. Lewanowicz, S., Keller, P., Woźny, P.: Bézier form of dual bivariate Bernstein polynomials. arxiv:1510.08246[math.NA] (2015)
12. Lewanowicz, S., Woźny, P.: Connections between two-variable Bernstein and Jacobi polynomials on the triangle. J. Comput. Appl. Math. **197**, 520–533 (2006)
13. Lewanowicz, S., Woźny, P., Keller, P.: Weighted polynomial approximation of rational Bézier curves. arXiv:1502.07877[math.NA] (2015)
14. Lewanowicz, S., Woźny, P., Keller, P.: Polynomial approximation of rational Bézier curves with constraints. Numer. Algor. **59**, 607–622 (2012)
15. Rivlin, T.J. Chebyshev Polynomials. From Approximation Theory to Algebra and Number Theory, 2nd edn. Wiley, New York (1990)
16. Sharma, R.: Conversion of a rational polynomial in triangular Bernstein-Bézier form to polynomial in triangular Bernstein-Bézier form. Internat. J. Comput. Appl. Math. **8**, 45–52 (2013)
17. Woźny, P., Lewanowicz, S.: Constrained multi-degree reduction of triangular Bézier surfaces using dual Bernstein polynomials. J. Comput. Appl. Math. **235**, 785–804 (2010)
18. Xu, H.X., Wang, G.J.: Approximating rational triangular Bézier surfaces by polynomial triangular Bézier surfaces. J. Comput. Appl. Math. **228**, 287–295 (2009)
19. Xu, H.X., Wang, G.J.: New algorithm of polynomial triangular B-B surfaces approximation to rational triangular B-B surfaces. J. Inform. Comput. Sci. **7**, 725–738 (2010)
20. Zhang, L., Wang, G.J.: An effective algorithm to approximate rational triangular B-B surfaces using polynomial forms (in Chinese). Chin. J. Comput. **29**, 2151–2162 (2006)