J. Yaghob (Ed.): ITAT 2015 pp. 30–37
Charles University in Prague, Prague, 2015

Efficient Computational Algorithm for Spline Surfaces

Lukáš Miňo

Institute of Computer Science, Faculty of Science, P. J. Šafárik University in Košice
Jesenná 5, 040 01 Košice, Slovakia
lukas.mino@upjs.sk

Abstract: Many data mining tasks can be reformulated as optimization problems, in the solution of which approximation by surfaces plays a key role. The paper proposes a new efficient computational algorithm for spline surfaces over uniform grids. The algorithm is based on a recent result on approximation of a biquartic polynomial by bicubic ones, that ensures C^2 continuity of the corresponding four bicubic spline components. As a consequence of this biquartic polynomial based approach to constructing spline surfaces, the classical de Boor's computational task breaks down to a reduced task and a simple remainder one. The comparison of the proposed and classical computational algorithm shows that the former needs less multiplication operations resulting in non negligible speed up.

1 Introduction

Recent years a considerable effort has been seen to develop reliable and efficient data mining tools to discover hidden knowledge in very large data bases. The fundamental problem is proposing algorithms to extract some useful information from very large databases. Fortunately, many data mining tasks can be reformulated as optimization problems, where approximation by surfaces plays a key role [1], [7].

The goal of the paper is to show that even such standard result as de Boor's sequential algorithm for construction of interpolating spline surfaces can be improved. It suggests a computational algorithm based on new model equations, in derivation of which biquartic polynomials played an essential role.

The idea of using quartic and biquartic polynomials in cubic and bicubic spline construction comes from recent results of Török and Szabó [16], [19], [13]. They have proven a key interrelation of these polynomials using the IZA representation [18], [14], which can incorporate both interpolation and approximation. The IZA representation was obtained using an r-point transformation that was a generalization of its three point ancestor [4]. A three point transformation was successfully applied to various approximation problems. Works [15], [8] showed how it can be used to assess the unknown degree in regression polynomials. In [5] a three point method was developed to detect piecewise cubic approximation segments for data with moderate errors. The technique, based on which the IZA representation has been derived, was first used in [10]. The paper [17] showed how to properly use the IZA representation's reference points for segment connection and their relation to derivatives. Papers [6], [14] contain results on approximation of 3D data based on the reference point approach. The first remarkable asymptotic properties of the IZA representation based two-part approximation model were gained in [11] and [18]. These properties confirmed the validity of the two-part approximation model, which led first to [16], where the interrelation of quartic and cubic polynomials was shown, and then to papers [19] and [13], [9] that introduced the reduced system approach to spline curve construction and proved the interrelation of bicubic and biquartic polynomials.

The algorithm presented in this paper has a decreased number of equations and is based on the generalized results of [19] and [9].

The structure of the article is as follows. Section two is devoted to problem statement. To be self-contained, section three briefly describes de Boor's algorithm. The next section first provides the definition of bicubic and biquartic polynomials. Then it shortly discusses the interrelation of these bivariate polynomials and their role in the computational schema. Section five contains the proposed sequential computational algorithm based on reduced systems. Section six briefly compares the new and the classical algorithm. The efficiency of the proposed algorithm is shown in the last but one section by computing the theoretical speedup that is approximately 1.33.

2 Problem Statement

The section defines the inputs for the spline surface, and the requirements that it should fulfil and based on which it can be constructed.

Consider a uniform grid

$$[u_0, u_1, \ldots, u_{2m}] \times [v_0, v_1, \ldots, v_{2n}], \tag{1}$$

where

$$u_i = u_0 + ih_x, \quad i = 1, 2, \ldots, 2m, m \in \mathbb{N},$$

$$v_j = v_0 + jh_y, \quad j = 1, 2, \ldots, 2n, n \in \mathbb{N}.$$

According to [3], the spline surface is defined by given values

$$z_{i,j}, \quad i = 0, 1, \ldots, 2m, \quad j = 0, 1, \ldots, 2n \tag{2}$$

at the equispaced grid-points, and given first directional derivatives

$$d^x_{i,j}, \quad i = 0, 2m, \quad j = 0, 1, \ldots, 2n \tag{3}$$

at boundary verticals,

$$d^y_{i,j}, \quad i = 0, 1, \ldots, 2m, \quad j = 0, 2n \tag{4}$$

at boundary horizontals and cross derivatives

$$d^{x,y}_{i,j}, \quad i = 0, 2m, \quad j = 0, 2n \tag{5}$$

at the four corners of the grid.

The task is to define a quadruple $[z_{i,j}, d^x_{i,j}, d^y_{i,j}, d^{x,y}_{i,j}]$ at every grid-point $[u_i, v_j]$, based on which a uniform bicubic clamped spline surface S of class C^2 can be constructed with properties

$$S(u_i, v_j) = z_{i,j}, \qquad \frac{\partial S(u_i, v_j)}{\partial y} = d^y_{i,j},$$
$$\frac{\partial S(u_i, v_j)}{\partial x} = d^x_{i,j}, \qquad \frac{\partial^2 S(u_i, v_j)}{\partial x \partial y} = d^{x,y}_{i,j},$$

where the adjacent spline segments are twice continuously differentiable. Our aim is to solve this task with less equations and less multiplications than the standard spline construction algorithm [3]. We will achieve this by means of Hermite splines and using a recently derived relationship property between biquartic and bicubic polynomials.

3 Carl de Boor's Algorithm

Paper [3] is devoted to bicubic spline surface interpolation. It formulates the problem and gives the solution to it. We briefly reformulate the paper's main result for uniform splines to show the four main equations, based on which the numerous tridiagonal systems of de Boor's algorithm for solution of the unknown derivatives are constructed. Thanks to it the paper is self contained and the reader can count up the number of operational multiplications and so quantitatively compare de Boor's and the proposed new algorithm.

Lemma 1 (de Boor)**.** *If the above z values and d derivatives are given, then the values*

$$\begin{aligned} &d^x_{i,j}, && i = 1, \ldots, 2m-1, \quad j = 0, \ldots, 2n, \\ &d^y_{i,j}, && i = 0, \ldots, 2m, \quad j = 1, \ldots, 2n-1, \\ &d^{x,y}_{i,j}, && i = 1, \ldots, 2m-1, \quad j = 0, \ldots, 2n, \\ & && \text{and } i = 0, \ldots, 2m, \quad j = 1, \ldots, 2n-1 \end{aligned}$$

are uniquely determined by the following $2(2m)+(2n)+5$ *linear systems of altogether* $3(2m)(2n)+(2m)+(2n)-5$ *equations:*
for $j = 0, \ldots, 2n$,

$$d^x_{i+1,j} + 4d^x_{i,j} + d^x_{i-1,j} = \frac{3}{h_x}(z_{i+1,j} - z_{i-1,j}), \tag{6}$$

where $i = 1, \ldots, 2m-1$;
for $j = 0, 2n$,

$$d^{x,y}_{i+1,j} + 4d^{x,y}_{i,j} + d^{x,y}_{i-1,j} = \frac{3}{h_x}(d^y_{i+1,j} - d^y_{i-1,j}), \tag{7}$$

where $i = 1, \ldots, 2m-1$;
for $i = 0, \ldots, 2m$,

$$d^y_{i,j+1} + 4d^y_{i,j} + d^y_{i,j-1} = \frac{3}{h_y}(z_{i,j+1} - z_{i,j-1}), \tag{8}$$

where $j = 1, \ldots, 2n-1$;
for $i = 0, \ldots, 2m$,

$$d^{x,y}_{i,j+1} + 4d^{x,y}_{i,j} + d^{x,y}_{i,j-1} = \frac{3}{h_y}(d^x_{i,j+1} - d^x_{i,j-1}), \tag{9}$$

where $j = 1, \ldots, 2n-1$.

4 Biquartic Polynomials and Bicubic Splines

The section begins with definition of bicubic and biquartic polynomials. Then it shortly discusses the interrelation of these bivariate polynomials and its role in the computational schema.

The tensor product formulas of bicubic Hermite spline components, see [12], and biquaric polynomials are given by the following two definitions.

Definition 1. *On grid (1) the bicubic Hermite spline components* $S_{i,j}(x,y)$ *for*

$$\begin{aligned} i = 0, 1, 2, \ldots, 2m-1, \qquad & j = 0, 1, 2, \ldots, 2n-1, \\ x \in [u_i, u_{i+1}], \qquad & y \in [v_j, v_{j+1}], \end{aligned}$$

are defined as follows

$$S_{i,j}(x,y) = \boldsymbol{\lambda}^T(x, u_i, h_x) \cdot \boldsymbol{\varphi}_{i,j} \cdot \boldsymbol{\lambda}(y, v_j, h_y), \tag{10}$$

where $\boldsymbol{\lambda}$ *is a vector of basis functions*

$$\boldsymbol{\lambda}(t, t_0, h) = \begin{bmatrix} \frac{(1+2\frac{t-t_0}{h})(t-t_1)^2}{h^2} \\ \frac{(t-t_0)^2(1-2\frac{t-t_1}{h})}{h^2} \\ \frac{(t-t_0)(t-t_1)^2}{h^2} \\ \frac{(t-t_0)^2(t-t_1)}{h^2} \end{bmatrix}^T,$$

$t_1 = t_0 + h$ *and* $\boldsymbol{\varphi}$ *is a matrix of function values and first derivatives*

$$\boldsymbol{\varphi}_{i,j} = \begin{pmatrix} z_{i,j} & z_{i,j+1} & d^y_{i,j} & d^y_{i,j+1} \\ z_{i+1,j} & z_{i+1,j+1} & d^y_{i+1,j} & d^y_{i+1,j+1} \\ d^x_{i,j} & d^x_{i,j+1} & d^{x,y}_{i,j} & d^{x,y}_{i,j+1} \\ d^x_{i+1,j} & d^x_{i+1,j+1} & d^{x,y}_{i+1,j} & d^{x,y}_{i+1,j+1} \end{pmatrix}.$$

For the spline components the following conditions hold

$$\begin{aligned}
S_{i,j}(u_k,v_l) &= z_{k,l}, \quad k=i,i+1, \quad l=j,j+1,\\
\frac{\partial S_{i,j}(u_k,v_l)}{\partial x} &= d^x_{k,l}, \quad k=i,i+1, \quad l=j,j+1,\\
\frac{\partial S_{i,j}(u_k,v_l)}{\partial y} &= d^y_{k,l}, \quad k=i,i+1, \quad l=j,j+1,\\
\frac{\partial^2 S_{i,j}(u_k,v_l)}{\partial x\partial y} &= d^{x,y}_{k,l}, \quad k=i,i+1, \quad l=j,j+1.
\end{aligned}$$

Based on (10) the second derivatives of $S_{i,j}(x,y)$ can be expressed effectively, e.g.

$$\frac{\partial^2 S_{i,j}(x,y)}{\partial x^2} = \frac{\partial^2 \boldsymbol{\lambda}^T(x,u_i,h_x)}{\partial x^2}\cdot\boldsymbol{\varphi}_{i,j}\cdot\boldsymbol{\lambda}(y,v_j,h_y), \quad (11)$$

where

$$\frac{\partial^2 \boldsymbol{\lambda}(t,t_0,h)}{\partial t^2} = \begin{bmatrix} \frac{6(2t-2t_0-h)}{h^3} \\ \frac{6(-2t+2t_0+h)}{h^3} \\ \frac{2(3t-3t_0-2h)}{h^2} \\ \frac{2(3t-3t_0-h)}{h^2} \end{bmatrix}^T.$$

The biquartic polynomials are also defined by tensor product.

Definition 2. *On grid (1) the biquartic polynomials $F_{i,j}(x,y)$ for*

$$\begin{aligned}
i=0,2,4,\ldots,2(m-1), \qquad & j=0,2,4,\ldots,2(n-1),\\
x\in[u_i,u_{i+2}], \qquad & y\in[v_j,v_{j+2}],
\end{aligned}$$

are defined as follows

$$F_{i,j}(x,y) = \mathbf{L}^{\mathbf{T}}(x,u_i,h_x)\cdot\boldsymbol{\Phi}_{i,j}\cdot\mathbf{L}(y,v_j,h_y), \quad (12)$$

where $\mathbf{L}$ is a vector of basis functions

$$\mathbf{L}(t,t_0,h) = \begin{bmatrix} \frac{-(1+2\frac{t-t_0}{h})(t-t_1)(t-t_2)^2}{4h^3} \\ \frac{(t-t_0)^2(t-t_2)^2}{h^4} \\ \frac{(t-t_0)^2(t-t_1)(1-2\frac{t-t_2}{h})}{4h^3} \\ \frac{-(t-t_0)(t-t_1)(t-t_2)^2}{4h^3} \\ \frac{(t-t_0)^2(t-t_1)(t-t_2)}{4h^3} \end{bmatrix}^T,$$

$t_1=t_0+h$, $t_2=t_0+2h$ and $\boldsymbol{\Phi}$ is a matrix of function values and first derivatives

$$\boldsymbol{\Phi}_{i,j} = \begin{pmatrix} z_{i,j} & z_{i,j+1} & z_{i,j+2} & d^y_{i,j} & d^y_{i,j+2} \\ z_{i+1,j} & z_{i+1,j+1} & z_{i+1,j+2} & d^y_{i+1,j} & d^y_{i+1,j+2} \\ z_{i+2,j} & z_{i+2,j+1} & z_{i+2,j+2} & d^y_{i+2,j} & d^y_{i+2,j+2} \\ d^x_{i,j} & d^x_{i,j+1} & d^x_{i,j+2} & d^{x,y}_{i,j} & d^{x,y}_{i,j+2} \\ d^x_{i+2,j} & d^x_{i+2,j+1} & d^x_{i+2,j+2} & d^{x,y}_{i+2,j} & d^{x,y}_{i+2,j+2} \end{pmatrix}.$$

For u_k, v_l defined in (1) the following conditions hold

$$\begin{aligned}
F_{i,j}(u_k,v_l) &= z_{k,l}, \quad k=j,j+1,j+2, \quad l=j,j+1,j+2,\\
\frac{\partial F_{i,j}(u_k,v_l)}{\partial x} &= d^x_{k,l}, \quad k=i,i+2, \quad l=j,j+1,j+2,\\
\frac{\partial F_{i,j}(u_k,v_l)}{\partial y} &= d^y_{k,l}, \quad k=i,i+1,i+2, \quad l=j,j+2,\\
\frac{\partial^2 F_{i,j}(u_k,v_l)}{\partial x\partial y} &= d^{x,y}_{k,l}, \quad k=i,i+2, \quad l=j,j+2.
\end{aligned}$$

The tensor product definition of $F_{i,j}(x,y)$ by (12) provides a compact way to express first derivatives, e.g.

$$\frac{\partial F_{i,j}(x,y)}{\partial y} = \mathbf{L}^{\mathbf{T}}(x,u_i,h_x)\cdot\boldsymbol{\Phi}_{i,j}\cdot\frac{\partial \mathbf{L}(y,v_j,h_y)}{\partial y}. \quad (13)$$

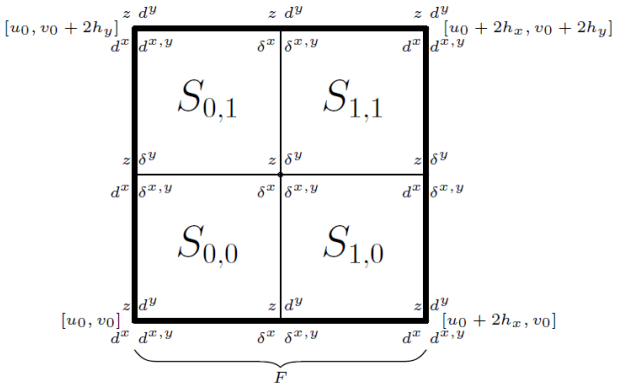

Figure 1: Schema of objects of a 2×2 - component bicubic Hermite spline surface.

Works [13], [9] prove how is a biquartic polynomial approximated by four bicubic polynomials. We want to apply this idea in our new approach to computing uniform bicubic splines of class C^2. The point of the approach is to solve only one half of derivatives from equations, and the second half of derivatives to compute from simple formulas that are derived from corresponding biquartic polynomials.

Unlike de Boor's lemma, we provide only the interpretation of the main result of [9]: a 2×2-component bicubic Hermite spline of class C^1 will be of class C^2, if the grid-points are equispaced and the unknown derivatives of the bicubic spline components at them are computed from a corresponding biquartic polynomial that is uniquely determined by the spline problem of Section 2 for the $[u_0,u_1,u_2]\times[v_0,v_1,v_2]$ grid.

This interrelation between a biquartic and four bicubic polynomials is illustrated by the schema in Fig. 1. The biquartic polynomial F over $[u_0,u_2]\times[v_0,v_2]$ is defined by given nine function values z and sixteen derivatives d that set up four quadruples $[z,d^x,d^y,d^{x,y}]$, two pairs $[z,d^x]$, two pairs $[z,d^y]$ and a single z. Every bicubic spline component is defined by four quadruples $[z,d^x,d^y,d^{x,y}]$. The nine quadruples in the figure are depicted around nine grid-points. Those eleven directional and cross first derivatives that are computed from the biquartic polynomial F and

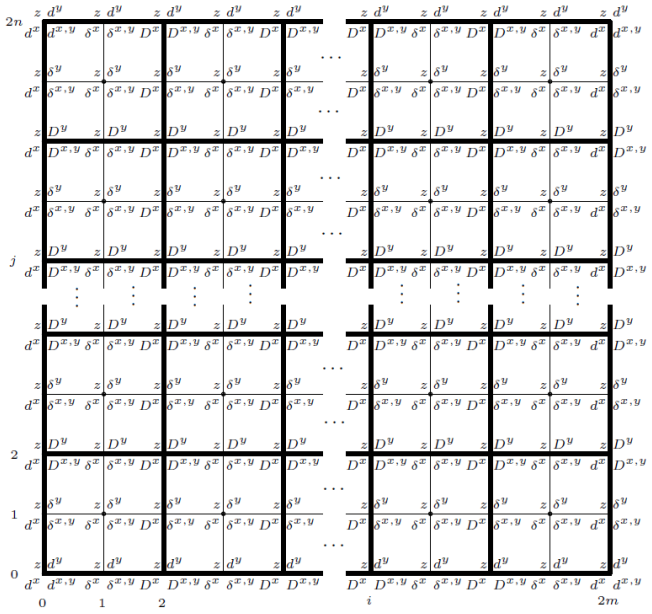

Figure 2: Schema of objects of a $(2m+1)\times(2n+1)$-component bicubic Hermite spline surface.

that are needed to construct the four bicubic spline components $S=\{S_{0,0},S_{0,1},S_{1,0},S_{1,1}\}$, are denoted by δ, see Fig. 1.

The below proposed algorithm was developed by generalizing the above described interrelation between biquartic and bicubic polynomials. First biquartic polynomials were handled and then based on them new model equations and formulas for unknown derivatives of the bicubic spline surface were derived.

Figure 2 illustrates the schema of the proposed computational algorithm for $(2m+1)\times(2n+1)$ bicubic spline surface of class C^2. There are $2m+1$ verticals and $2n+1$ horizontals. Rectangles and thick rectangles indicate the boundary of bicubic spline components and biquartic polynomials, respectively. There are two types of objects at every grid-point: known and unknown ones. The given values and derivatives are denoted by z, d and the unknown first derivatives by D, δ. Notice that z is provided at every grid-point and d only along the total grid's boundary. The most important is where are the unknown D and δ parameters. The D parameters are located only along the thick rectangles, but never in their center. As we shall see later the D parameters will be computed from equations and the δ parameters from explicit formulas. The equations were derived from the equality of second derivatives of spline components and the formulas from the biquartic polynomials.

The derived new model equations for the unknown D derivatives of the spline surface segments and parts of the explicite formulas for δ are generalization of model equations and formulas of the unknown derivatives of spline curve segments, see [16].

5 Reduced System Algorithm

This section presents a new sequential algorithm for computing a C^2-class uniform spline surface's unknown first derivatives. Its efficiency will be shown in the next section. The central part of the algorithm are three new model equations and five new explicit formulas. We do not derive these model equations and explicit formulas, only mention that for their derivation we had to (see Fig. 2) thoroughly analyse the structure of the bicubic and biquartic polynomials, specify which derivatives should be the D and which the δ parameters, understand which polynomials are critical for obtaining the equations and formulas, and for all this the following steps were needed:

1. construction of some biquartic polynomials $F_{i,j}$, see (12),
2. construction of δ parameters as functions, see (13),
3. construction of some appropriate Hermite spline components S, see (10), for comparing of their second derivatives, see (11).

The below proposed algorithm can be characterized from two aspects

- what it computes,
- the quality of its outcome.

The algorithm computes D and δ coefficients for bicubic spline surface components from inputs given at equispaced grid-points described in Section 2. The D coefficients are computed from linear systems based on equations (14), (16), (18) – (21). The δ coefficients are gained from explicit formulas (15), (17), (22) – (24). Since the equations for the D parameters were derived from the equality of second derivatives of spline components and the formulas for the δ parameters were gained from biquartic polynomials that as we know grant C^2 continuity of their components, the algorithm provides such coefficients that the uniform bicubic spline surface will be of class C^2.

Algorithm for computing the unknown first derivatives of the spline surface in three main steps with reduced systems.
Inputs: z and d values, see (2) – (5).
Step 1a. Computation of D^x parameters along the horizontals from equation systems.

For each horizontal we construct a system of linear equations to compute the D^x values, located on the inside odd grid-points. Each horizontal represents an independent tridiagonal system of linear equations.

For each horizontal, see Fig. 2, $j=0,1,\ldots,2n$, a tridiagonal system is constructed based on equations

$$D^x_{2(i+1),j}-14D^x_{2i,j}+D^x_{2(i-1),j}= \\ =\frac{3}{h_x}(z_{2(i+1),j}-z_{2(i-1),j})-\frac{12}{h_x}(z_{2i+1,j}-z_{2i-1,j}), \tag{14}$$

where $i=1,2,\ldots,m-1$.

Step 1b. Computation of δ^x parameters from explicit formulas.

To finish the computation of all first partial derivatives with respect to x we have to calculate

$$\delta^x_{i,j} = \frac{3}{4h_x}(z_{i+1,j} - z_{i-1,j}) - \frac{1}{4}(d^x_{i+1,j} + d^x_{i-1,j}), \quad (15)$$

where $i = 1,3,\ldots,2m-1$, $j = 1,3,\ldots,2n-1$.

Step 2a. Computation of D^y parameters along the horizontals from equation systems.

For each vertical we construct a system of linear equations to compute the D^y values, located on the inside odd grid-points. Each vertical represents an independent system of linear equations.

For each vertical, $i = 0,1,\ldots,2m$, a tridiagonal system is constructed based on equations

$$\begin{aligned} &D^y_{i,2(j+1)} - 14D^y_{i,2j} + D^y_{i,2(j-1)} = \\ &= \frac{3}{h_y}(z_{i,2(j+1)} - z_{i,2(j-1)}) - \frac{12}{h_y}(z_{i,2j+1} - z_{i,2j-1}), \end{aligned} \quad (16)$$

where $j = 1,2,\ldots,n-1$.

Step 2b. Computation of δ^y parameters from explicit formulas

To finish the computation of all first partial derivatives with respect to y we have to calculate

$$\delta^y_{i,j} = \frac{3}{4h_y}(z_{i,j+1} - z_{i,j-1}) - \frac{1}{4}(d^y_{i,j+1} + d^y_{i,j-1}), \quad (17)$$

where $i = 1,3,\ldots,2m-1$, $j = 1,3,\ldots,2n-1$.

At this moment all directional derivatives are known: some were provided and the unknown D and δ directional ones were computed in Steps 1 and 2. In the further steps all directional derivatives will be denoted by d and contained in the right hand side of equations and formulas.

Step 3a. Computation of $D^{x,y}$ parameters along the bottom and top horizontals, and left vertical from equation systems.

We construct systems of linear equations for bottom and top horizontals and left verticals by [3]. The systems for the bottom boundary horizontal is

$$D^{x,y}_{i+1,0} + 4D^{x,y}_{i,0} + D^{x,y}_{i-1,0} = \frac{3}{h_x}(d^y_{i+1,0} - d^y_{i-1,0}), \quad (18)$$

where $i = 1,\ldots,2m-1$;
for the top boundary horizontal is

$$D^{x,y}_{i+1,2n} + 4D^{x,y}_{i,2n} + D^{x,y}_{i-1,2n} = \frac{3}{h_x}(d^y_{i+1,2n} - d^y_{i-1,2n}), \quad (19)$$

where $i = 1,\ldots,2m-1$;
and for the left boundary vertical is

$$D^{x,y}_{0,j+1} + 4D^{x,y}_{0,j} + D^{x,y}_{0,j-1} = \frac{3}{h_y}(d^x_{0,j+1} - d^x_{0,j-1}), \quad (20)$$

where $j = 1,2,\ldots,2n-1$.

Step 3b. Computation of $D^{x,y}$ parameters from the inside grid-points using systems of equations

For the odd verticals, $i = 2,4,6,\ldots,2m$, a tridiagonal system is constructed based on equations

$$\begin{aligned} &D^{x,y}_{i,j+2} - 14D^{x,y}_{i,j} + D^{x,y}_{i,j-2} = \\ &= \frac{1}{7}(d^{x,y}_{i-2,j+2} + d^{x,y}_{i-2,j-2}) - 2d^{x,y}_{i-2,j} + \\ &+ \frac{3}{7h_x}(d^y_{i-2,j+2} + d^y_{i-2,j-2}) + \frac{3}{7h_y}(-d^x_{i-2,j+2} + d^x_{i-2,j-2}) + \\ &+ \frac{9}{7h_x}(d^y_{i,j+2} + d^y_{i,j-2}) + \frac{9}{7h_xh_y}(-z_{i-2,j+2} + z_{i-2,j-2}) + \\ &+ \frac{12}{7h_x}(-d^y_{i-1,j+2} - d^y_{i-1,j-2}) + \frac{12}{7h_y}(d^x_{i-2,j+1} - d^x_{i-2,j-1}) + \\ &+ \frac{3}{h_y}(d^x_{i,j+2} - d^x_{i,j-2}) + \frac{27}{7h_xh_y}(-z_{i,j+2} + z_{i,j-2}) + \\ &+ \frac{36}{7h_xh_y}(z_{i-1,j+2} - z_{i-1,j-2} + z_{i-2,j+1} - z_{i-2,j-1}) - \\ &- \frac{6}{h_x}d^y_{i-2,j} + \frac{12}{h_y}(d^x_{i,j+1} + d^x_{i,j-1}) + \frac{108}{7h_xh_y}(z_{i,j+1} - z_{i,j-1}) - \\ &- \frac{18}{h_x}d^y_{i,j} + \frac{144}{7h_xh_y}(-z_{i-1,j+1} + z_{i-1,j-1}) + \frac{24}{h_x}d^y_{i-1,j}, \end{aligned} \quad (21)$$

where $j = 4,6,\ldots,2n-4$.

Mention must be made, that this step was the most critical. At first after computation of $D^{x,y}$ unknowns along the bottom and top horizontals in Step 3a we got equations with six $D^{x,y}$ unknowns on the left side. Török suggested to compute the Dxy parameters along the left vertical using de Boors equation in Step 3a and thanks to this three of six $D^{x,y}$ parameters could be moved to the right side as computed.

Step 3c. Computation of $\delta^{x,y}$ parameters from explicit formulas

To finish the computation of all first cross derivatives we have to calculate for the even verticals and the even horizontals

$$\begin{aligned} \delta^{x,y}_{i,j} = &\frac{1}{16}(d^{x,y}_{i+1,j+1} + d^{x,y}_{i+1,j-1} + d^{x,y}_{i-1,j+1} + d^{x,y}_{i-1,j-1}) - \\ &- \frac{3}{16h_y}(d^x_{i+1,j+1} - d^x_{i+1,j-1} + d^x_{i-1,j+1} - d^x_{i-1,j-1}) - \\ &- \frac{3}{16h_x}(d^y_{i+1,j+1} + d^y_{i+1,j-1} - d^y_{i-1,j+1} - d^y_{i-1,j-1}) + \\ &+ \frac{9}{16h_xh_y}(z_{i+1,j+1} - z_{i+1,j-1} - z_{i-1,j+1}z_{i-1,j-1}), \end{aligned} \quad (22)$$

where $i = 1,3,\ldots,2m-1$, $j = 1,3,\ldots,2n-1$;
for the even verticals and the odd horizontals

$$\delta^{x,y}_{i,j} = \frac{3}{4h_y}(d^x_{i,j+1} - d^x_{i,j-1}) - \frac{1}{4}(d^{x,y}_{i,j+1} + d^{x,y}_{i,j-1}), \quad (23)$$

where $i = 1,3,\ldots,2m-1$, $j = 2,4,\ldots,2(n-1)$;
and for the odd verticals and the even horizontals

$$\delta^{x,y}_{i,j} = \frac{3}{4h_y}(d^x_{i,j+1} - d^x_{i,j-1}) - \frac{1}{4}(d^{x,y}_{i,j+1} + d^{x,y}_{i,j-1}), \quad (24)$$

where $i = 2,4,\ldots,2m-1$, $j = 1,3,\ldots,2n-1$.

6 The Comparision of the New and de Boor's Algorithm

We sum up the complete spline task and how is it completed by the considered two algorithms. Then we give some details about the role of the biquartic polynomials that are absent in de Boor's algorithm but played a crucial task in the design of the proposed one.

In case of $(2m+1)(2n+1)$ grid-points, to fulfill the complete spline task means to construct $(2m)(2n)$ spline components using $(2m+1)(2n+1)$ various quadruples, where one quadruple looks the following way: $[z, d^x, d^y, d^{x,y}]$. The input comprises $(2m+1)(2n+1)$ z values, $2(2n+1)$ d^x, $2(2m+1)$ d^y and 4 $d^{x,y}$ derivatives. The unknown derivatives require computation.

In de Boor's algorithm every unknown derivative is computed from equations that are of four types, see (6)–(9) and [3]:

- from $2n+1$ systems with $2m-1$ variables $(2n+1)(2m-1)$ derivatives d^x are computed,
- from 2 systems with $2m-1$ variables $2(2m-1)$ derivatives $d^{x,y}$ are computed,
- from $2m+1$ systems with $2n-1$ variables $(2m+1)(2n-1)$ derivatives d^y are computed,
- from $2m+1$ systems with $2n-1$ variables $(2m+1)(2n-1)$ derivatives $d^{x,y}$ are computed.

The proposed algorithm's benefit is that lesser number of unknown derivatives (D parameters) are computed from systems of equations, see Tab. 3, compared to de Boor's algorithm [3]. The remaining derivatives (δ parameters) are computed from explicit formulas. This was achieved thanks to using mn biquartic polynomials $F_{i,j}(x,y)$ behind the scene, whose definitions use only $(m+1)(n+1)$ quadruples.

The algorithm's drawback is that it uses more types of relation: six types of equations and five types of explicit formulas. Nevertheless it has to compute approximately $12/5$ times less equations within systems – compare Tab. 2 and Tab. 3 from Section 7.

Let us have a closer look at biquartic polynomials and their role in the algorithm's design. One biquartic polynomial $F_{i,j}(x,y)$ needs 25 parameters, see Definition 2. We distinguish between four types of F polynomials, see Fig. 2,

1. at the corners,
2. at boundary horizontals,
3. at boundary verticals,
4. over the inside grid-points.

While for example for the biquartic polynomial F over inside grid-points all the sixteen derivatives are unknown, they are D parameters, for F from the corners only seven are unknown and nine are given, these are the d parameters. After obtaining the D parameters the remaining derivatives are computed based on F. From every F eleven delta parameters can be obtained: two pairs of type $[\delta^x, \delta^{x,y}]$, two pairs of type $[\delta^y, \delta^{x,y}]$ and one triple $[\delta^x, \delta^y, \delta^{x,y}]$. Naturally, δ parameters are functions of D parameters, see [9].

After introducing the new algorithm in the previous section and giving a short insight into its design in this one, the next section is devoted to its quantitative characterization.

7 Number of Multiplications

The standard way of solving a tridiagonal linear system

$$\underbrace{\begin{bmatrix} b & 1 & 0 & & & \\ 1 & b & 1 & & & \\ 0 & 1 & b & & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & & & & b \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_K \end{bmatrix}}_{\boldsymbol{d}} = \underbrace{\begin{bmatrix} r_1 - d_0 \\ r_2 \\ r_3 \\ \vdots \\ r_K - d_{K+1} \end{bmatrix}}_{\boldsymbol{r}}$$

uses the LU factorization $\boldsymbol{A}\boldsymbol{d} = \boldsymbol{L}\underbrace{\boldsymbol{U}\boldsymbol{d}}_{\boldsymbol{y}} = \boldsymbol{r}$, where

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & & & & \\ l_2 & 1 & & & & \\ 0 & l_3 & 1 & & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & & l_K & 1 \end{bmatrix},$$

$$\boldsymbol{U} = \begin{bmatrix} u_1 & 1 & 0 & & & \\ 0 & u_1 & 1 & & & \\ & & u_2 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & u_K \end{bmatrix},$$

the u_i and l_i elements are computed as, see [2],

$$\boldsymbol{LU}: \quad u_1 = b, \{l_i = \frac{1}{u_{i-1}}, u_i = b - l_i\}, i = 2, ..., K, \quad (25)$$

and the forward (Fw) and backward (Bw) steps of the solution are

$$\text{Forward:} \quad \boldsymbol{L}\boldsymbol{y} = \boldsymbol{r}, \quad (26)$$

where $y_1 = r_1$, $\{y_i = r_i - l_i y_{i-1}\}$, $i = 2, \ldots, K$;

$$\text{Backward:} \quad \boldsymbol{U}\boldsymbol{d} = \boldsymbol{y}, \quad (27)$$

where $d_K = \frac{y_K}{u_K}$, $\{d_i = \frac{1}{u_i}(y_i - x_{i+1})\}$, $i = K-1, \ldots, 1$.

The tridiagonal systems of equations for de Boor's and our algorithm are solved by LU decomposition. All the systems of these algorithms are diagonally dominant with

	(25)	(26)	(27)		
Dim.	**LU**	**Fw**	**Bw**	**RHS**	**Total mult.**
$N \times N$	γN	N	γN	βN	$2\gamma N + \beta N + N$

Table 1: Count of multiplications in one system of equations

de Boor	**System**	**Equation**	β
Step 1 (6)	$2n+1$	$2m-1$	1
Step 2 (7)	2	$2m-1$	1
Step 3 (8)	$2m+1$	$2n-1$	1
Step 4 (9)	$2m+1$	$2n-1$	1
Total equations	$12mn+2m+2n-5$		
Total mult.	$24\gamma mn+24mn+4\gamma m+4\gamma n$ $+4m+4n-10\gamma-1$		

Table 2: Count based characteristics – de Boor's algorithm

Proposed	**System**	**Equation**	β
Step 1a (14)	$2n+1$	$m-1$	2
Step 2a (16)	$2m+1$	$n-1$	2
Step 3a (18)	1	$2m-1$	1
Step 3a (19)	1	$2m-1$	1
Step 3a (20)	1	$2n-1$	1
Step 3b (21)	m	$n-1$	17
Total equations	$5mn+2m+n-5$		
Total mult.	$10\gamma mn+30mn+4\gamma m+2\gamma n$ $-13m+n-10\gamma-12$		

Table 3: Count based characteristics – proposed algorithm

Proposed	**Formula**	β
Step 1b (15)	$m(2n+1)$	2
Step 2b (17)	$(2m+1)n$	2
Step 3c (22)	mn	4
Step 3c (23)	$m(n-1)$	2
Step 3c (24)	mn	2
Total formulas	$7mn+n$	
Total mult.	$16mn+2n$	

Table 4: Count based characteristics – explicit formulas in proposed algorithm

elements 1, 4, 1 and 1, -14, 1. The LU and backward steps contain a division that is indicated by γ, the ratio between division and multiplication: the performance of a division operation is equivalent to γ multiplications.

The proposed and de Boor's algorithm differ

- in number of systems of equations,
- in number of equations within systems,
- in number of multiplication operations on right hand side (RHS) of equations.

Tab. 1 presents the number of multiplications for solving one general tridiagonal $N \times N$ matrix, where β denotes the number of multiplications on the right hand side of an equation.

The second and third columns of Tab. 2 and Tab. 3 provide the count of equations within the given steps (equations) and the count of equations within a system, respectively, for a grid of size $(2m+1) \times (2n+1)$. The last but one rows contain the total number of equations. The total count of multiplications to solve the tridiagonal systems within de Boor's and the proposed algorithm are in the last row.

In the proposed algorithm we evaluate in addition to the solution of the tridiagonal systems of equations, see Tab. 3, as well as δ parameters using explicit formulas. Therefore the total number of multiplications in the proposed algorithm based on tables 3 and 4 is

$$10\gamma mn + 46mn + 4\gamma m + 2\gamma n - 13m + 3n - 10\gamma - 12.$$

We can see that the number of multiplication in the proposed algorithm is less. The theoretical speed up for some various grid sizes were computed under assumption that $\gamma = 3.5$. Based on Tab. 5 we can conclude that the proposed model is asymptotically 1.33 times faster.

Grid	**de Boor**	**Proposed**	**Speed up**
11×11	2 835	2 033	1.394
101×101	271 755	203 003	1.339
1001×1001	27 017 955	20 255 453	1.333
$(10^6+1) \times (10^6+1)$	$27 \cdot 10^{12}$	$20.25 \cdot 10^{12}$	1,333
$(10^{12}+1) \times (10^{12}+1)$	$27 \cdot 10^{24}$	$20.25 \cdot 10^{24}$	1,333

Table 5: Speed up

8 Conclusion

We suggested a new efficient sequential algorithm for computation of a spline surface over an equispaced grid. Its theoretically evaluated asymptotic speed up over de Boors algorithm is approximately 1.33. The algorithm has also a very nice property from the view point of parallel computation: the computation of the second half of unknowns based on explicit equations can be parallelized automatically. Naturally, parallel methods, suggested for solving tridiagonal systems of de Boor's algorithm can be used for solution of the new algorithm's tridiagonal systems as well. Therefore, the proposed reduced system based algorithm is preferable over de Boor's algorithm not only for sequential, but also for parallel computation.

Acknowledgement

The author thanks Cs. Török for the problem statement, his helpful suggestions and support. This work was partially supported by the research grants VEGA 1/0073/15 and VVGS-PF-2015-477.

References

[1] Berry M. J., Linoff G. S.: Data mining techniques (Third Edition). For Marketing, Sales, and Customer Relationship Management, John Wiley and Sons, 2011

[2] Björck A.: Numerical methods in matrix computations. Springer, 2015

[3] de Boor C.: Bicubic spline interpolation. Journal of Mathematics and Physics, **41 (3)** (1962), 212–218

[4] Dikoussar N. D.: Function parametrization by using 4-point transforms. Comput. Phys. Commun. **99** (1997), 235–254

[5] Dikoussar N. D., Török C.: Automatic Knot Finding for Piecewise Cubic Approximation. Mat. Model., 2006, T-17, N.3

[6] Dikoussar N. D., Török C.: Kybernetika **43 (4)** (2007), 533–546

[7] Hegland M., Roberts S., Altas I.: Finite element thin plate splines for data mining applications. In: M. Daehlen, T. Lyche, and L. Schumaker, (eds.), Mathematical Methods for Curves and Surfaces II, pages 245–252, Nashville, TN, 1998. Vanderbilt University Press. Available as Mathematical Research Report MRR 057-97, School of Mathematical Sciences, Australian National University

[8] Matejčiková A., Török C.: Noise suppression in RDPT. Forum Statisticum Slovacum, 3/2005, Bratislava, ISSN 1336-7420, 199–203

[9] Miňo L., Szabó I., Török C.: Bicubic splines and biquartic polynomials, 2015, to appear

[10] Révayová M., Török C.: Piecewise approximation and neural networks. Kybernetika **43 (4)** (2007), 547–559

[11] Révayová M., Török C.: Reference points based recursive approximation. Kybernetika **49 (1)** (2013), 60–72

[12] Salomon D.: Curves and surfaces for computer graphics. Springer, 2006

[13] Szabó I.: Approximation algorithms for 3D data analysis. PhD Thesis, P. J. Šafárik University in Košice, Slovakia, 2015, to appear

[14] Szabó I., Török C.: Smoothing in 3D with reference points and Polynomials. 29th Spring Conference on Computer Graphics SCCG 2013, Smolenice–Bratislava, Comenius University, 39–43

[15] Török C.: 4-point transforms and approximation, Comput. Phys. Commun. **125** (2000), 154–166

[16] Török C.: On reduction of equations' number for cubic splines. Matematicheskoe modelirovanie **26 (11)** (2014), 33–36

[17] Török C.: Piecewise smoothing using shared parameters. Forum Statisticum Slovacum, 7/2009, 188–193

[18] Török C.: Reference points based transformation and approximation. Kybernetika **49 (4)** (2013), `http://www.kybernetika.cz/content/2013/4/644/paper.pdf`

[19] Török C.: Speedup of interpolating spline construction, 2015, to appear