E11-2015-77

L. Miňo [1,*], Cs. Török [1,**]

FAST ALGORITHM FOR SPLINE SURFACES

[1] Institute of Computer Science, Faculty of Science, P. J. Šafárik University in Košice, Jesenná 5, 040 01 Košice, Slovakia

[*] E-mail: lukas.mino@upjs.sk

[**] E-mail: csaba.torok@upjs.sk

Миньо Л., Торок Ч. E11-2015-77
Быстрый алгоритм вычисления сплайновых поверхностей

Предлагается быстрый алгоритм для вычисления параметров интерполирующих сплайновых поверхностей с непрерывностью C^2 на равномерных сетках, использующий недавно доказанное свойство между биквартическими и бикубическими многочленами, благодаря которому размер трехдиагональных систем уменьшается наполовину. Сравнение предложенного алгоритма с классическим алгоритмом де Бура показывает, что первый требует меньшего числа умножений и его явные формулы можно распараллелить автоматически.

Работа выполнена в Лаборатории информационных технологий ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 2015

Miňo L., Török Cs. E11-2015-77
Fast Algorithm for Spline Surfaces

We propose a fast algorithm for evaluation of interpolating spline surfaces with C^2 continuity over uniform grids that utilizes a recently proved approximation property between biquartic and bicubic polynomials. Thanks to this property, the size of the tridiagonal systems are reduced to the half. The comparison of the proposed algorithm with the classical de Boor's one shows that the former needs less multiplication operations and its explicit formulas can be parallelized automatically.

The investigation has been performed at the Laboratory of Information Technologies, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 2015

1. INTRODUCTION

Surfaces play a key role in many fields, such as data modelling, computational physics or computer graphics. The paper improves the standard de Boor's sequential algorithm [2] for construction of interpolating spline surfaces based on new model equations, in derivation of which biquartic polynomials played an essential role. The proposed algorithm results in reduced systems and less number of multiplications.

The idea of using quartic and biquartic polynomials in uniform cubic and bicubic spline construction, respectively, comes from [9], where the interrelation of quartic and cubic polynomials within a two-part approximation model was proven. The reduced system approach to spline curve construction was introduced in paper [10].

Paper [5] showed the validity of the 2×2-part approximation model with bicubic and biquartic polynomials. The algorithm with decreased number of equations presented in this paper generalizes the results of [10] and [5] to surfaces.

The structure of the article is as follows. Section 2 is devoted to problem statement and to a short description of the spline order's influence on the smoothness of the spline surface. The next section discusses the interrelation of the bicubic and biquartic polynomials and their role in the computational schema. Section 4 contains the proposed sequential computational algorithm based on reduced systems. The efficiency of the proposed algorithm is shown in the last but one section by computing the theoretical speedup that is approximately 1.4. The Appendix provides the definition of bicubic and biquartic polynomials.

2. PROBLEM STATEMENT

The section defines the inputs for a spline surface based on which it can be constructed. The influence of the spline order on the smoothness of the spline surfaces is discussed shortly, too.

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 [2], 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,

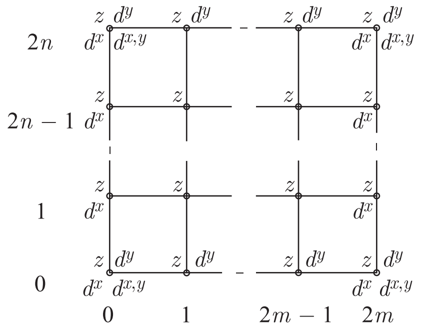
$$d^y_{i,j}, \quad i = 0, 1, \ldots, 2m, \quad j = 0, 2n \tag{4}$$

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.

Figure 1 depicts the schema of the inputs: directional derivatives d^x, d^y are given only on the boundaries and at the inner grid-points there are only function values z.



z d^y z d^y z d^y z d^y
$2n$ d^x $d^{x,y}$ d^x $d^{x,y}$
z z z z
$2n-1$ d^x d^x
z z z z
1 d^x d^x
z d^y z d^y z d^y z d^y
0 d^x $d^{x,y}$ d^x $d^{x,y}$
0 1 $2m-1$ $2m$

Fig. 1. Inputs for a uniform bicubic clamped spline of C^2

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 $[u_i, v_j]$, where

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

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

in such a way that the resulting uniform bicubic clamped spline surface S will be of class C^2, i.e., the adjacent spline segments will be twice continuously differentiable. Our aim is to solve this task with less equations and less multiplications than the standard spline construction algorithm [2]. We will achieve this by means of Hermite splines and using a recently derived relationship property between biquartic and bicubic polynomials.

Let as have a short look at the differences between bicubic spline surfaces of class C^1 and C^2. Hermite spline surfaces, see Appendix, are by default of class C^1, see [3,6,8]. It is sufficient to take any finite values for the quadruples at the grid-points and the spline will be automatically of class C^1 due to its definition. However, the derivatives within the computed quadruples can be selected in such a way that the second derivatives of the adjacent surface segments will be also continuous and so the resulting bicubic spline will be of class C^2.

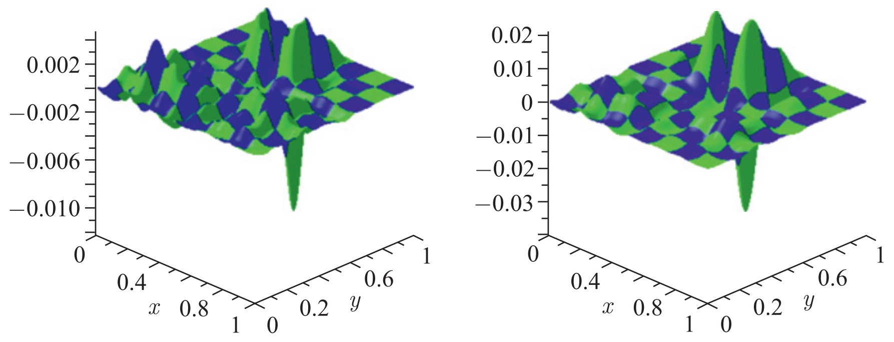


Fig. 2. Differences of the Franke function and the spline surfaces of classes C^1 and C^2

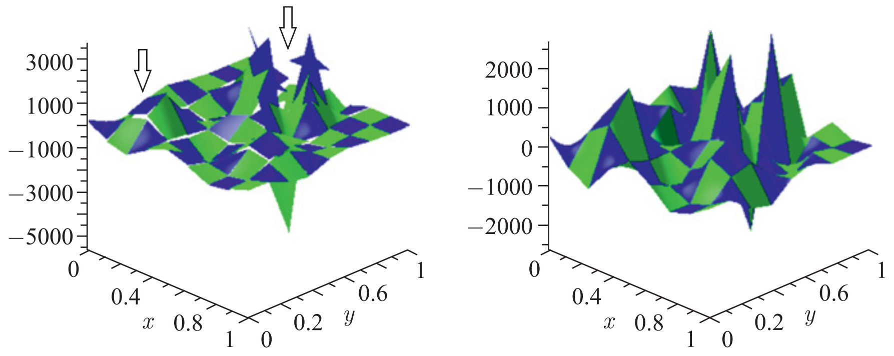


Fig. 3. Second derivatives of the spline surfaces of classes C^1 and C^2

Whether to use C^1 or C^2 bicubic spline surfaces depends on the character of the solving problem. The peaks and valleys can be better approximated with C^1 spline surfaces; however, the second derivatives of such a spline will be discontinuous. Figure 2 illustrates the differences of the Franke function and the corresponding interpolating spline surfaces of class C^1 (on the left) and C^2 (on the right) over a 9×9 grid. The differences on the left are approximately four times smaller. Figure 3 contains the second derivatives of the spline surfaces. Notice that the second derivatives of the spline surface of class C^1 are not continuous. So, to have less bumpy, more smooth surface for modelling data with continuous second derivatives, choose a model based on C^2 splines.

3. BIQUARTIC POLYNOMIALS AND BICUBIC SPLINES

The section discusses shortly the interrelation of bicubic and biquartic polynomials and gives 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.

Works [5,7] prove how a biquartic polynomial is 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 to compute the second half of derivatives from simple formulas that are derived from corresponding biquartic polynomials.

We provide here the interpretation of the main result of [5]: 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. 4. 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 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.

The algorithm proposed below 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.

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 $[z, d^x, d^y, d^{x,y}]$. 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.

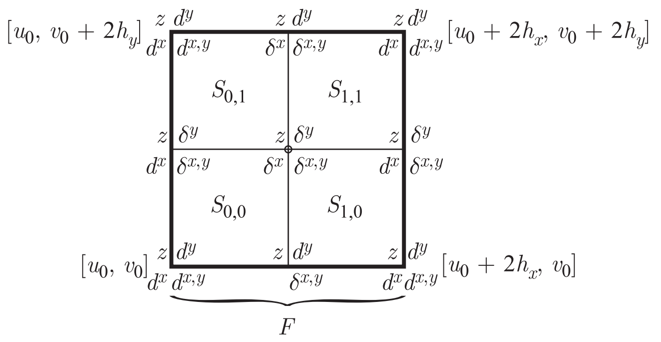

$[u_0,\ v_0 + 2h_y]$
z d^y d^x $d^{x,y}$
z d^y δ^x $\delta^{x,y}$
z d^y d^x $d^{x,y}$
$[u_0 + 2h_x,\ v_0 + 2h_y]$
$S_{0,1}$
$S_{1,1}$
z δ^y d^x $\delta^{x,y}$
z δ^y δ^x $\delta^{x,y}$
z δ^y d^x $\delta^{x,y}$
$S_{0,0}$
$S_{1,0}$
$[u_0,\ v_0]$
z d^y d^x $d^{x,y}$
z d^y $\delta^{x,y}$
z d^y d^x $d^{x,y}$
$[u_0 + 2h_x,\ v_0]$
F

Fig. 4. Schema of objects of a 2×2-component bicubic Hermite spline surface

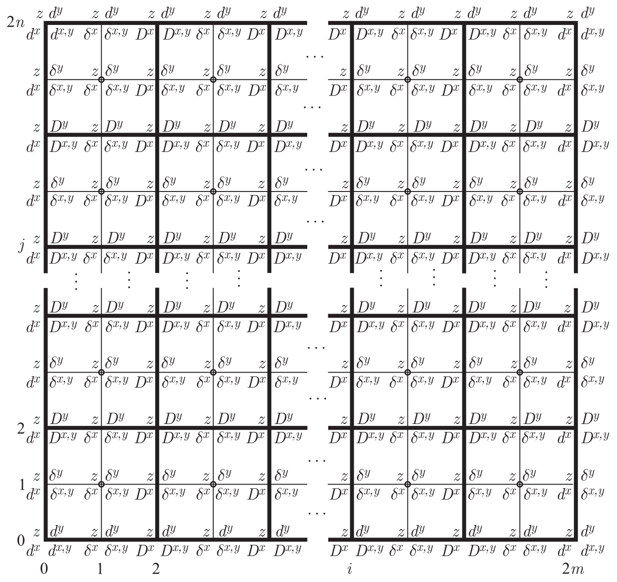


Fig. 5. Schema of objects of a $(2m + 1) \times (2n + 1)$-component bicubic Hermite spline surface

Figure 5 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 grid-points: 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 the unknown D and δ parameters are. 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.

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. 5:

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 [5].

The derived new model equations for the unknown D derivatives of the spline surface segments and parts of the explicit formulas for δ are generalization of model equations and formulas of the unknown derivatives of spline curve segments, see [9].

4. REDUCED SYSTEM ALGORITHM

This section presents a new sequential algorithm for computing the unknown first derivatives of a C^2-class uniform bicubic spline surface. Its efficiency will be shown in the next section. The central part of the algorithm is 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. 5) 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 (22),

2) construction of δ parameters as functions, see (23),

3) construction of some appropriate Hermite spline components S, see (20), for comparing of their second derivatives, see (21).

The algorithm proposed below 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 (6), (8), (10)–(13). The δ coefficients are gained from explicit formulas (7), (9), (14)–(16). 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 a C^2-class uniform bicubic 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. 5, $j = 0, 1, \dots, 2n$, a tridiagonal system is constructed based on equations

$$D^x_{2(i+1),j} - 14 D^x_{2i,j} + D^x_{2(i-1),j} = \\ = \frac{3}{h_x}(z_{2(i+1),j} - z_{2(i-1),j}) - \frac{12}{h_x}(z_{2i+1,j} - z_{2i-1,j}), \quad (6)$$

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

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

$$\delta^x_{i,j} = \frac{3}{4h_x}(z_{i+1,j} - z_{i-1,j}) - \frac{1}{4}(d^x_{i+1,j} + d^x_{i-1,j}), \quad (7)$$

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

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, $i = 0, 1, \dots, 2m$, a tridiagonal system is constructed based on equations

$$D^y_{i,2(j+1)} - 14 D^y_{i,2j} + D^y_{i,2(j-1)} = \\ = \frac{3}{h_y}(z_{i,2(j+1)} - z_{i,2(j-1)}) - \frac{12}{h_y}(z_{i,2j+1} - z_{i,2j-1}), \quad (8)$$

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

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

$$\delta^y_{i,j} = \frac{3}{4h_y}(z_{i,j+1} - z_{i,j-1}) - \frac{1}{4}(d^y_{i,j+1} + d^y_{i,j-1}), \quad (9)$$

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

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 [2]; therefore, the left-hand sides of equations contain 4 instead of -14. The system for the bottom boundary horizontal is

$$D^{x,y}_{i+1,0} + 4D^{x,y}_{i,0} + D^{x,y}_{i-1,0} = \frac{3}{h_x}(d^y_{i+1,0} - d^y_{i-1,0}), \tag{10}$$

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

$$D^{x,y}_{i+1,2n} + 4D^{x,y}_{i,2n} + D^{x,y}_{i-1,2n} = \frac{3}{h_x}(d^y_{i+1,2n} - d^y_{i-1,2n}), \tag{11}$$

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

$$D^{x,y}_{0,j+1} + 4D^{x,y}_{0,j} + D^{x,y}_{0,j-1} = \frac{3}{h_y}(d^x_{0,j+1} - d^x_{0,j-1}), \tag{12}$$

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

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

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

$$\begin{aligned} D^{x,y}_{i,j+2} - 14D^{x,y}_{i,j} + D^{x,y}_{i,j-2} &= \frac{1}{7}(d^{x,y}_{i-2,j+2} + d^{x,y}_{i-2,j-2}) - 2d^{x,y}_{i-2,j} + \\ &+ \frac{3}{7h_x}(d^y_{i-2,j+2} + d^y_{i-2,j-2}) + \frac{3}{7h_y}(-d^x_{i-2,j+2} + d^x_{i-2,j-2}) + \\ &+ \frac{9}{7h_x}(d^y_{i,j+2} + d^y_{i,j-2}) + \frac{9}{7h_xh_y}(-z_{i-2,j+2} + z_{i-2,j-2}) + \\ &+ \frac{12}{7h_x}(-d^y_{i-1,j+2} - d^y_{i-1,j-2}) + \frac{12}{7h_y}(d^x_{i-2,j+1} - d^x_{i-2,j-1}) + \\ &+ \frac{3}{h_y}(d^x_{i,j+2} - d^x_{i,j-2}) + \frac{27}{7h_xh_y}(-z_{i,j+2} + z_{i,j-2}) + \\ &+ \frac{36}{7h_xh_y}(z_{i-1,j+2} - z_{i-1,j-2} + z_{i-2,j+1} - z_{i-2,j-1}) - \\ &- \frac{6}{h_x}d^y_{i-2,j} + \frac{12}{h_y}(d^x_{i,j+1} + d^x_{i,j-1}) + \frac{108}{7h_xh_y}(z_{i,j+1} - z_{i,j-1}) - \\ &- \frac{18}{h_x}d^y_{i,j} + \frac{144}{7h_xh_y}(-z_{i-1,j+1} + z_{i-1,j-1}) + \frac{24}{h_x}d^y_{i-1,j}, \end{aligned} \tag{13}$$

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

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 in the left side. If we compute the $D^{x,y}$ parameters along the left vertical using de Boor's equations, as was done in Step 3a, then three of six $D^{x,y}$ parameters can 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

$$\begin{aligned}\delta^{x,y}_{i,j} = \frac{1}{16}(d^{x,y}_{i+1,j+1} + d^{x,y}_{i+1,j-1} + d^{x,y}_{i-1,j+1} + d^{x,y}_{i-1,j-1}) - \\ - \frac{3}{16h_y}(d^{x}_{i+1,j+1} - d^{x}_{i+1,j-1} + d^{x}_{i-1,j+1} - d^{x}_{i-1,j-1}) - \\ - \frac{3}{16h_x}(d^{y}_{i+1,j+1} + d^{y}_{i+1,j-1} - d^{y}_{i-1,j+1} - d^{y}_{i-1,j-1}) + \\ + \frac{9}{16h_xh_y}(z_{i+1,j+1} - z_{i+1,j-1} - z_{i-1,j+1}z_{i-1,j-1}),\end{aligned} \tag{14}$$

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

$$\delta^{x,y}_{i,j} = \frac{3}{4h_y}(d^{x}_{i,j+1} - d^{x}_{i,j-1}) - \frac{1}{4}(d^{x,y}_{i,j+1} + d^{x,y}_{i,j-1}), \tag{15}$$

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

$$\delta^{x,y}_{i,j} = \frac{3}{4h_y}(d^{x}_{i,j+1} - d^{x}_{i,j-1}) - \frac{1}{4}(d^{x,y}_{i,j+1} + d^{x,y}_{i,j-1}), \tag{16}$$

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

The proposed algorithm's benefit is that lesser number of unknown derivatives (D parameters) are computed from systems of equations, see further in Table 2, compared to de Boor's algorithm [2]. 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 relations than de Boor's algorithm. While the latter uses four types of model equations, our algorithm six types of model equations and five types of explicit formulas. Nevertheless, it has to compute approximately $12/5$ times less equations within systems, see Table 4.

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

5. NUMBER OF MULTIPLICATIONS

The tridiagonal linear systems of equations for de Boor's and the proposed algorithm are diagonally dominant with elements 1, 4, 1 and 1, -14, 1. One of the standard ways of solving a tridiagonal linear system

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

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

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & & & \\ l_2 & 1 & & & \\ 0 & l_3 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots \\ & & & l_K & 1 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} u_1 & 1 & 0 & & \\ 0 & u_1 & 1 & & \\ & & u_2 & & \\ & & \ddots & \ddots & \ddots \\ & & & & u_K \end{bmatrix},$$

the u_i and l_i elements are computed as, see [1],

$$\boldsymbol{L}\,\boldsymbol{U}: \; u_1 = b, \; \left\{ l_i = \frac{1}{u_{i-1}}, \; u_i = b - l_i \right\}, \; i = 2, \dots, K, \tag{17}$$

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

$$\text{Forward: } \boldsymbol{L}\,\boldsymbol{y} = \boldsymbol{r}, \; y_1 = r_1, \; \{ y_i = r_i - l_i y_{i-1} \}, \; i = 2, \dots, K; \tag{18}$$

$$\text{Backward: } \boldsymbol{U}\,\boldsymbol{d} = \boldsymbol{y}, \; d_K = \frac{y_K}{u_K}, \; \left\{ d_i = \frac{1}{u_i}(y_i - x_{i+1}) \right\}, \; i = K-1, \dots, 1. \tag{19}$$

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 algorithms differ in

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

Table 1. **Count of multiplications in one system of equations**

Dim.	LU (17)	Fw (18)	Bw (19)	RHS	Total mult.
$K \times K$	γK	K	γK	βK	$2\gamma K + \beta K + K$

Table 1 presents the number of multiplications for solving one general tridiagonal system with a $K \times K$ matrix, where β denotes the number of multiplications in the right-hand side of an equation.

The second and third columns of Table 2 provide the count of equations within the given algorithmic steps (equations) and the count of equations within a system, respectively, for a grid of size $(2m + 1) \times (2n + 1)$.

Table 2. **Count based characteristics – proposed algorithm**

Proposed	System	Equation
Step 1a (6)	$2n + 1$	$m - 1$
Step 2a (8)	$2m + 1$	$n - 1$
Step 3a (10)	1	$2m - 1$
Step 3a (11)	1	$2m - 1$
Step 3a (12)	1	$2n - 1$
Step 3b (13)	m	$n - 1$

In the proposed algorithm we evaluate in addition to the solution of the reduced tridiagonal systems of equations the δ parameters using explicit formulas, see Table 3.

Table 3. **Count based characteristics – explicit formulas in proposed algorithm**

Proposed	Formula
Step 1b (7)	$m(2n + 1)$
Step 2b (9)	$(2m + 1)n$
Step 3c (14)	mn
Step 3c (15)	$m(n - 1)$
Step 3c (16)	mn

Table 4 shows how the total count of multiplications in the reduced and remainder tasks of the algorithm were gained from the total count of equations and formulas, respectively. For comparison the table contains also the number of multiplications of de Boor's algorithm, see [4]. We can see from it that

Table 4. **The total count of multiplications**

Algorithm	de Boor	Proposed
Equations	$12mn + 2m + 2n - 5$	$5mn + 2m + n - 5$
Formulas		$7mn + n$
Mult. in equations	$24\gamma mn + 24mn +$ $+4\gamma m + 4\gamma n +$ $+4m + 4n - 10\gamma - 1$	$10\gamma mn + 30mn +$ $+4\gamma m + 2\gamma n -$ $-13m + n - 10\gamma - 12$
Mult. in formulas		$16mn + 2n$

the number of multiplications in the proposed algorithm is less. The theoretical speedup is given by the ratio

$$\frac{24\gamma mn + 24mn}{10\gamma mn + 46mn},$$

which is equivalent under assumption that $\gamma \approx 4$ to

$$\frac{96mn + 24mn}{40mn + 46mn} = \frac{120}{86} \doteq 1.4.$$

Mention must be made that the proposed algorithm has also a very nice property from the viewpoint of parallel computation: the explicit formulas based calculation of the second half of unknowns within the remainder tasks of the algorithm can be parallelized automatically, because the formulas are independent of each other. Naturally, parallel methods, suggested for solving tridiagonal systems, can be used for solution of our tridiagonal systems as well. Therefore, the proposed reduced system based algorithm may be preferable to de Boor's algorithm not only for sequential, but also for parallel computation.

6. CONCLUSION

The paper suggested a new efficient sequential algorithm for bicubic spline surfaces over an equispaced grid utilizing biquartic polynomials. As a consequence of this biquartic polynomial based approach to constructing spline surfaces, the total computational task broke down to reduced tasks and remainder ones, where the latter compute the bigger part of unknown derivatives using simple explicit formulas. The comparison of the proposed and classical computational algorithm shows that the former needs less multiplication operations resulting in non-negligible speedup.

APPENDIX

The tensor product formulas of bicubic Hermite spline components, see [6], 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*

$$i = 0,1,2,\dots,2m-1, \quad j = 0,1,2,\dots,2n-1,$$
$$x \in [u_i, u_{i+1}], \quad y \in [v_j, v_{j+1}]$$

are defined as follows:

$$S_{i,j}(x,y) = \boldsymbol{\lambda}^{\boldsymbol{T}}(x,u_i,h_x)\cdot\boldsymbol{\varphi}_{i,j}\cdot\boldsymbol{\lambda}(y,v_j,h_y), \tag{20}$$

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

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

$t_1 = t_0 + h$ 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 (20) the second derivatives of $S_{i,j}(x,y)$ can be expressed effectively, e.g.,

$$\frac{\partial^2 S_{i,j}(x,y)}{\partial x^2} = \frac{\partial^2 \boldsymbol{\lambda}^T(x,u_i,h_x)}{\partial x^2} \cdot \boldsymbol{\varphi}_{i,j} \cdot \boldsymbol{\lambda}(y,v_j,h_y), \tag{21}$$

where

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

The biquartic polynomials are also defined by tensor product.

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

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

are defined as follows:

$$F_{i,j}(x,y) = \mathbf{L}^{\mathbf{T}}(x,u_i,h_x) \cdot \boldsymbol{\Phi}_{i,j} \cdot \mathbf{L}(y,v_j,h_y), \tag{22}$$

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

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

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

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

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 (22) provides a compact way to express first derivatives, e.g.,

$$\frac{\partial F_{i,j}(x, y)}{\partial y} = \mathbf{L}^{\mathbf{T}}(x, u_i, h_x) \cdot \mathbf{\Phi}_{i,j} \cdot \frac{\partial \mathbf{L}(y, v_j, h_y)}{\partial y}. \tag{23}$$

REFERENCES

1. Björck A.: Numerical Methods in Matrix Computations, Springer, 2015.
2. de Boor C.: Bicubic Spline Interpolation, Journal of Mathematics and Physics, 41(3), 1962, P. 212–218.
3. Dikoussar N. D., Török Cs.: On One Approach to Local Surface Smoothing, Kybernetika, 43 (4), 2007, P. 533–546.
4. Miňo L.: Efficient Computational Algorithm for Spline Surfaces, 2015, to appear.
5. Miňo L., Szabó I., Török Cs.: Bicubic Splines and Biquartic Polynomials, 2015, to appear.
6. Salomon D.: Curves and Surfaces for Computer Graphics, Springer, 2006.

7. Szabó I.: Approximation Algorithms for 3D Data Analysis, PhD Thesis, P. J. Šafárik University in Košice, Slovakia, 2015, to appear.

8. Szabó I., Török Cs.: Smoothing in 3D with Reference Points and Polynomials, 29th Spring Conference on Computer Graphics SCCG 2013, Smolenice–Bratislava, Comenius University, ISBN 9788022333771, P. 39–43.

9. Török Cs.: On Reduction of Equations' Number for Cubic Splines, Matematicheskoe modelirovanie, vol. 26 (2014), no. 11, ISSN 0234-0879, P. 33–36.

10. Török Cs.: Speedup of Interpolating Spline Construction, 2015, to appear.

Received on August 17, 2015.

Редактор *Е. И. Кравченко*

Подписано в печать 12.10.2015.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 1,19. Уч.-изд. л. 1,61. Тираж 245 экз. Заказ № 58650.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@jinr.ru
www.jinr.ru/publish/