Scattered Data Interpolation with Quartic Triangular Bézier Patches: An Optimized Implementation

Krassimira Vlachkova[0000−0002−2159−6953] and Krum Radev

Abstract We present a new program package for interactive modeling and visualization of smooth scattered data interpolation using quartic triangular Bézier patches (TBP). We implemented an optimized algorithm (Vlachkova, 2021) based on quartic smooth interpolation curve networks and splitting. The algorithm allows to reduce the complexity of the resulting surface and to improve its smoothness. We have chosen the open-source data visualization library `Plotly.js` as our main implementation and visualization tool. This choice ensures the platform independency of our package and its direct use without restrictions. The package can be used for experiments with user's data sets since it works with the host file system. The latter allows wide testing, modeling, and editing of the resulting interpolation surfaces. We performed a large number of experiments using data of increasing complexity. Here we present and comment the results of our work.

1 Introduction

Interpolation of data points in $\mathbb{R}^3$ by smooth surface is a fundamental problem in applied mathematics which finds applications in a variety of fields such as medicine, architecture, archeology, computer graphics and animation, and more. Recently, the problem has become particularly relevant in bioinformatics and scientific visualization for surface reconstruction.

We consider the following problem: Given a set of points $(x_i, y_i, z_i) \in \mathbb{R}^3$, $i = 1, \ldots, n$, find a bivariate function $F(x, y)$ defined in a certain domain D containing points $V_i = (x_i, y_i)$, such that F possesses continuous partial derivatives up to a given order, and $F(x_i, y_i) = z_i$.

Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", 5 James Bourchier Blvd., 1164 Sofia, Bulgaria
e-mail: krassivl@fmi.uni-sofia.bg
e-mail: kvradev@uni-sofia.bg

Various methods and approaches for solving this problem were proposed and discussed, see, e.g., the survey papers [10, 9, 4, 19, 16], and also [6, 3]. A standard approach to solve the problem consists of two steps, see [10]:

1. Construct a triangulation $T = T(V_1, \ldots, V_N)$;

2. For every triangle in T construct a surface which interpolates the data in the three vertices.

The interpolation surface constructed in Step 2 is usually polynomial or piecewise polynomial. Typically, the patches are computed with a priori prescribed normal vectors at the data points. G^1 or G^2 smoothness of the resulting surface is achieved either by increasing the degree of the patches, or by the so called *splitting* [5] in which for each triangle in T a macro-patch consisting of a fixed number of Bézier sub-patches is constructed. Splitting was originally proposed by Clough and Tocher [5] and further developed by Percell [12] and Farin [7] for solving different problems. Splitting allows to keep the degree of the Bézier patches low by increasing the degrees of freedom. In practice using patches of least degree and splitting is preferable since it is computationally simple and efficient. Known splitting algorithms apply splitting to all macro-patches defined in D. We point out that splitting decreases the smoothness of a polynomial macro-patch to G^1.

Shirman and Séquin [13, 15] construct a G^1 smooth surface consisting of quartic TBP. Their method assumes that the normal vectors at the data points are given as part of the input. Shirman and Séquin construct a smooth cubic curve network defined on the edges of T, first, and then degree elevate it to quartic. Next, they apply splitting where for each triangle in T a macro-patch consisting of three quartic Bézier sub-patches is constructed. The interpolation surfaces constructed by Shirman and Séquin's algorithm often suffer from unwanted bulges, tilts, and shears as pointed out by the authors in [14] and more recently by Hettinga and Kosinka in [8].

Nielson [11] proposes a method which computes a smooth interpolation curve network defined on the edges of T so as to have common tangent planes at the data points and to satisfy an extremal property. This curve network is called *minimum norm network* (MNN) and is cubic. Nielson extends the MNN to a smooth interpolation surface using a *blending* method based on convex combination schemes. The interpolant obtained is a rational function on every triangle in T.

Vlachkova and Radev [18] propose an algorithm for interpolation of data in $\mathbb{R}^3$ which improves on Shirman and Séquin's approach in the following way. First, they use Nielson's MNN which is degree elevated to quartic curve network. A significant advantage of Nielson's method is that the normal vectors at the data points are obtained through the computation of the MNN. Second, they extend the MNN to a smooth surface consisting of quartic TBP so that to avoid unwanted distortions and twists which appear in surfaces constructed by Shirman and Séquin's method. As a result, the quality of the resulting surfaces is improved.

Vlachkova [17] proposes an improved algorithm for interpolation of data in $\mathbb{R}^3$ using quartic TBP which does not necessarily apply splitting to all triangles of T. The algorithm identify an optimized subset of triangles in T where splitting needs to

be applied. This reduces the complexity of the resulting surfaces and increases their smoothness.

In this paper we present a new software package for interactive visualization, manipulation, and comparison of smooth interpolation surfaces consisting of quartic TBP. Our work and contributions are in the field of experimental algorithmics and algorithm engineering. We implemented the optimized algorithm proposed in [17]. Our package is user friendly and has a number of features that allow easy testing and experimenting. We performed a large number of experiments. The results were analyzed and compared to reveal the main differences between our algorithm [18] and the optimized algorithm [17] with respect to various criteria including smoothness and shape of the resulting surfaces. Moreover, the experiments performed provide useful information on how to modify the surfaces constructed using the optimized algorithm [17] to further improve their shape and quality.

The paper is organized as follows. In Section 2 we briefly describe the three algorithms used, namely Nielson's MNN, the algorithm [18], and the optimized algorithm [17]. Detailed information about the implementation and visualization tool chosen and the main features of the package is provided in Section 3. In Section 4 we present the results from our experimental work. Our conclusions are made in the final Section 4.

2 Preliminaries and related work

2.1 Nielson's MNN

Let $n \geq 3$ be an integer and $P_i := (x_i, y_i, z_i),\ i = 1, \ldots, n$ be different points in $\mathbb{R}^3$. We call this set of points *data*. We assume that the projections $V_i := (x_i, y_i)$ of the data into the plane Oxy are different and non-collinear. Emphasizing the fact that V_i are in a general position, although not necessarily irregularly placed, we call such data *scattered*.

Definition 1 A collection of non-overlapping, non-degenerate triangles in Oxy is a *triangulation* of the points $V_i,\ i = 1, \ldots, n$, if the set of the vertices of the triangles coincides with the set of the points $V_i,\ i = 1, \ldots, n$.

Hereafter we assume that a triangulation T of the points $V_i,\ i = 1, \ldots, n$, is given and fixed. Furthermore, for the sake of simplicity, we assume that the domain D formed by the union of the triangles in T is connected. In general D is a collection of polygons with holes. The set of the edges of the triangles in T is denoted by E. If there is an edge between V_i and V_j in E, it will be referred to by e_{ij} or simply by e if no ambiguity arises.

Definition 2 A *curve network* is a collection of real-valued univariate functions $\{f_e\}_{e \in E}$ defined on the edges in E.

With any real-valued bivariate function F defined on D we naturally associate the curve network defined as the restriction of F on the edges in E, i. e. for $e = e_{ij} \in E$,

$$\begin{gathered} f_e(t) := F\Big(\big(1 - \frac{t}{\|e\|}\big)x_i + \frac{t}{\|e\|}\, x_j,\ \big(1 - \frac{t}{\|e\|}\big)y_i + \frac{t}{\|e\|}\, y_j\Big), \\ \text{where } 0 \le t \le \|e\| \text{ and } \|e\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \end{gathered} \tag{1}$$

Furthermore, according to the context F will denote either a real-valued bivariate function or a curve network defined by (1). We introduce the following class of *smooth interpolants* defined on D

$$\mathcal{F} := \big\{F(x,y) \in C^1(D) \mid F(x_i, y_i) = z_i,\ i = 1, \dots, N, f_e' \in AC,\ f_e'' \in L^2,\ e \in E\big\},$$

where $C^1(D)$ is the class of bivariate functions defined in D which possess continuous first order partial derivatives, AC is the class of univariate absolutely continuous functions defined in $[0, \|e\|]$, and L^2 is the class of univariate functions defined in $[0, \|e\|]$ whose second power is Lebesgue integrable.

The restrictions on E of the functions in $\mathcal{F}$ form the corresponding class of so-called *smooth interpolation curve networks*

$$C(E) := \big\{F_{|E} = \{f_e\}_{e\in E} \mid F(x,y) \in \mathcal{F}\big\}.$$

The smoothness of the interpolation curve network $F \in C(E)$ geometrically means that at each point P_i there is a *tangent plane* to F, where a plane is *tangent* to the curve network at the point P_i if it contains the tangent vectors at P_i of the curves incident to P_i.

For $F \in C(E)$ we denote the curve network of second derivatives of F by $F'' := \{f_e''\}_{e\in E}$. The L^2-norm of F'' is defined by

$$\|F''\|_{L^2(T)} := \|F''\| = \left(\sum_{e\in E} \int_0^{\|e\|} |f_e''(t)|^2 dt\right)^{1/2}.$$

Nielson [11] considered and solved the following extremal problem

$$(\mathbf{P}) \quad \textit{Find } F^* \in C(E) \ \textit{ such that } \|F^{*\prime\prime}\| = \inf_{F\in C(E)} \|F''\|.$$

The unique solution (MNN) to (**P**) is a cubic curve network and is obtained by solving a linear system of equations.

2.2 Two algorithms for scattered data interpolation using quartic TBP

Here we briefly present two algorithms for scattered data interpolation using quartic TBP. Although we use the algorithms with the degree elevated MNN as input, we

note that both algorithms work also with arbitrary quartic smooth interpolation curve networks.

Algorithm 1 below was proposed in [18] and improves on Shirman and Séquin's algorithm in various ways. Its input is the degree elevated MNN. Algorithm 1 takes a triangle in T and the degree-elevated quartic boundary control points of the corresponding macro-patch and computes 19 control points of the three G^1-continuous quartic Bézier sub-patches, see Fig. 1.

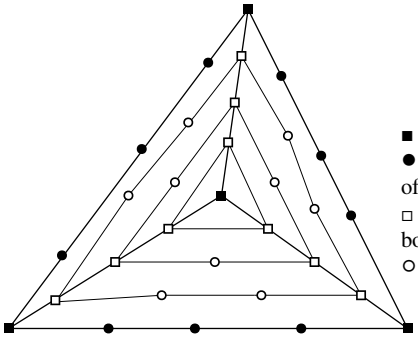

Fig. 1 Construction of a G^1-continuous Bézier macro-patch by splitting to three sub-patches.

Algorithm 1 [18]

Step 1. Compute the control points in the first layer:
1.1 Points of type □ are centers of the three small triangles with vertices ●■●.
1.2 Then points of type ○ are computed as described in [18].
Step 2. Compute the control points in the second layer:
2.1 Points of type □ are centers of the three small triangles with vertices ○□○ in the first layer.
2.2 Then points of type ○ are mid-points of the segments with vertices of type □ in the second layer.
Step 3. Compute the control points in the third layer: The three points of type □ are centers of the small triangles with vertices ○□○ in the second layer.
Step 4. Compute the splitting point of type ■ as a center of the triangle with vertices □ in the third layer.

Algorithm 2 was proposed in [17]. It improves on Algorithm 1 since it reduces the complexity of the resulting surface and increases its smoothness. First, Algorithm 2 represents all cubic curves comprising the MNN as quartic Bézier curves using degree elevation. Next, it identifies an optimized set T_s of triangles where splitting needs to be applied. For the remaining triangles, those in $T \setminus T_s$, it simply constructs the standard quartic Bézier patch.

Algorithm 2 below takes the MNN as an input and construct a G^1-continuous interpolating surface $F(x, y)$ defined on D which consists of triangular quartic Bézier patches.

Algorithm 2 [17]

Step 1. Compute the control points of the curves in the MNN.
Step 2. Degree elevate all curves to quartic curves.
Step 3. Compute an optimized set T_s as described in [17].
Step 4. For each $\tau \in T \setminus T_s$ compute 3 inner control points and construct a macro-patch.
Step 5. For each $\tau \in T_s$ compute 19 inner control points and construct three sub-patches using Algorithm 1 for splitting.

3 Overview of our work

We implemented a program package for interactive 3D modeling, visualization, and comparison of smooth interpolation surfaces consisting of quartic TBP. Our main goal was to implement Algorithm 2 proposed in [17] for constructing a G^1-smooth piecewise quartic surface that interpolates a given smooth quartic curve network. We need to visualize the proven mathematical properties of the resulting surface, and to compare them with the surfaces obtained by some known methods such as Shirman and Séquin's and with our Algorithm 1. For this purpose, a computer software for 3D visualization capable of displaying curves and TBP viewed from different angles and coloured in different colours should be created. Hence, the software needs to be interactive, easily configured as it runs, and allowing dynamic loading of input data. Since Algorithm 2 involves many mathematical calculations, mostly on points in $\mathbb{R}^3$, it would be convenient to display the coordinates of these points on the screen, for example when clicking the mouse, or to be able to hide and display individual parts of the surface. At the same time the software must be fast enough to work in real time and to visualize more complex surfaces. Taking into account the above requirements, we decided to design our software as a Web application. We used only HTML and JavaScript for the implementation. This decision enabled the resulting surfaces to be easily rendered using various visualization libraries. We have chosen `Plotly` [1] as our main implementation and visualization tool. This software is open source and is distributed under MIT license [2]. Using `Plotly` provides conveniences such as displaying the coordinate system, the coordinates of points when hovering over them, and controlling the objects rendered. There are, however, some significant disadvantages: few settings for the light sources and mostly very slow operation which makes Plotly not suitable for rendering more complex surfaces.

In the implementation the mouse is used to change the camera position. The surfaces can be edited interactively by rotation and resizing. The user interface for the program package is shown in Fig. 2.

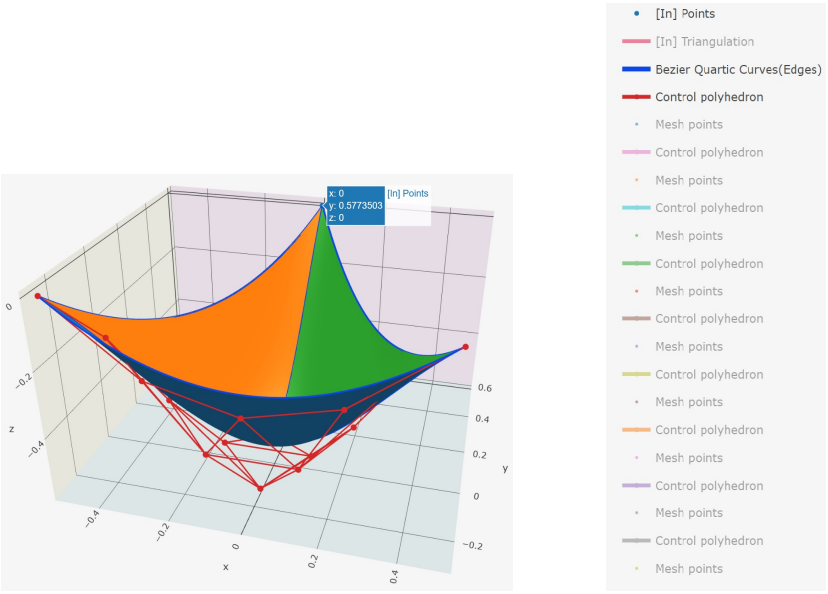

Fig. 2 The user interface for the program package created with `Plotly`.

4 Results from the experiments

Here we present and comment on two examples from the experiments performed and we share our observations and conclusions. In both examples we use the corresponding MNN which is degree elevated to quartic.

Example 1 We consider data obtained from a symmetric triangular pyramid. We have $n = 4$, $V_1 = (-1/2, -\sqrt{3}/6)$, $V_2 = (1/2, -\sqrt{3}/6)$, $V_3 = (0, \sqrt{3}/3)$, $V_4 = (0, 0)$, and $z_i = 0, i = 1, 2, 3$, $z_4 = -1/2$. The corresponding triangulation is shown in Fig. 3 (left). The MNN is shown in Fig. 3 (right). The surface generated using Algorithm 1 where all four macro-patches are splitted is shown in 4 (left). The surface generated using the optimized Algorithm 2 has no splitted macro-patches and is shown Fig. 4 (right). In this case the optimized set T_s is the empty set.

Example 2 We consider the convex function $f = \exp\left((x-0.5)^2 + (y-0.5)^2\right)$ which is sampled at 25 points shown in Table 1. The triangulation, which is shown in Fig. 5 (left), is the Delaunay triangulation and consists of 40 triangles. The corresponding MNN is shown in Fig. 5 (right). The surface generated using Algorithm 1 where all 40 macro-patches are splitted is shown in Fig. 4 (left). The surface generated using the optimized Algorithm 2 is shown Fig. 4 (right). The optimized set T_s is marked in black and consists of 6 triangles.

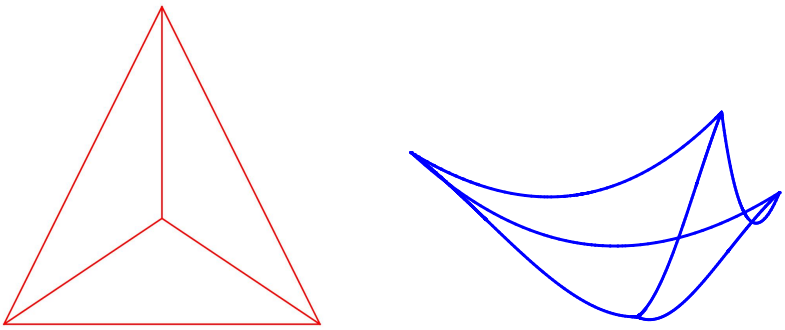

Fig. 3 Example 1: (left) The triangulation; (right) The MNN

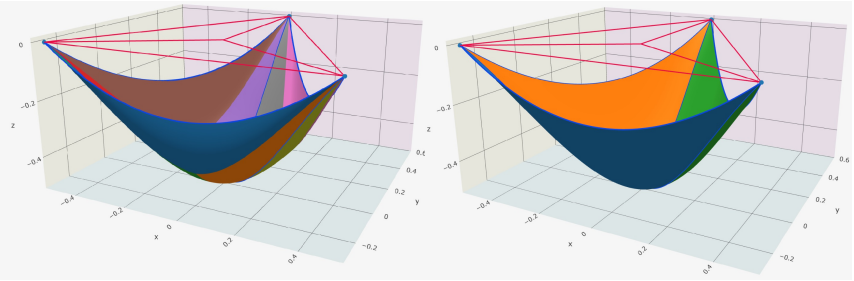

Fig. 4 Example 1. The surface interpolating the degree elevated MNN: (left) The surface generated using Algorithm 1 where all macro-patches are splitted; (right) The surface generated using the optimized Algorithm 2 where no macro-patch is splitted.

Table 1 The data for Example 2

i	1	2	3	4	5	6	7	8	9	10	11	12	13
x_i	.21	.46	.83	.97	.67	.53	.28	.07	.06	.25	.48	.67	.77
y_i	.88	.93	.89	.54	.71	.74	.77	.70	.43	.56	.61	.54	.45
i	14	15	16	17	18	19	20	21	22	23	24	25	
x_i	.90	.66	.50	.32	.25	.46	.57	.75	.94	.46	.18	.14	
y_i	.31	.35	.47	44	.31	.33	.20	.25	.05	.07	.19	.06	

5 Conclusion and future work

In this paper we presented a program package for interactive implementation and 3D visualization of smooth surfaces consisting of quartic TBP that interpolate given smooth curve network defined on the edges of an associated triangulation. The package are made user friendly and possess a wide range of features that allow experimenting with various data set and curve networks. Given the importance of

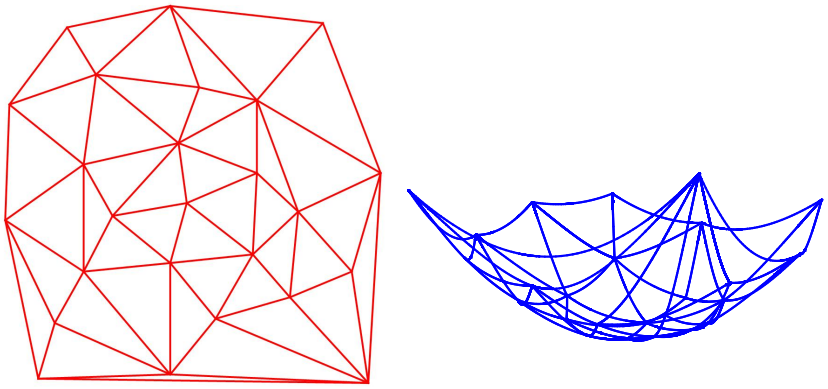

Fig. 5 Example 1: (left) The Delaunay triangulation; (right) The MNN

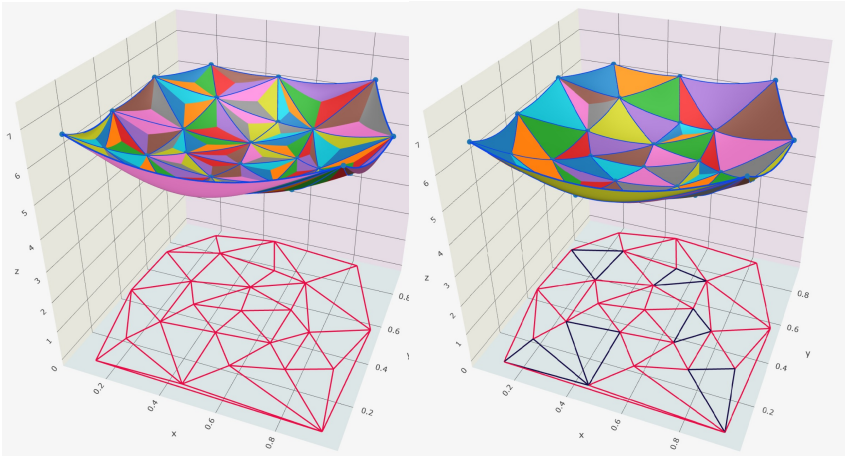

Fig. 6 Example 2: The surface interpolating the degree elevated MNN: (left) The surface generated using Algorithm 1 where all macro-patches are splitted; (right) The surface generated using the optimized Algorithm 2. The optimized set T_s is marked in black and consists of 6 triangles.

surface modeling and simulation techniques in practice, it is important to better understand and engineer software packages based on various criteria and algorithms for surface manipulation. We plan to further expand the package by adding new features to it, e.g. visualizing the highlight lines and gaussian curvature of the surfaces displayed.

Acknowledgements This work was supported in part by Sofia University Science Fund Grant No. 80-10-103/2023, and European Regional Development Fund and the Operational Program "Science and Education for Smart Growth" under contract № BG05M2OP001-1.001-0004 (2018-2023).

References

1. Plotly.js. https://plot.ly/javascript (last accessed April 22, 2023)
2. MIT License. https://opensource.org/licenses/MIT (last accessed August 22, 2022)
3. Anjyo, K., Lewis, J., Pighin, F.: Scattered data interpolation for computer graphics, SIGGRAPH 2014 course notes (2014). URL http://olm.co.jp/rd/research_event/scattered-data-interpolation-for-computer-graphics. Last accessed November 19, 2022
4. Berger, M., Tagliasacchi, A., Seversky, L., Alliez, P., Guennebaud, G., Levine, J., Sharf, A., Silva, C.: A survey of surface reconstruction from point clouds. Comput. Graph. Forum **36**(1), 301–329 (2017). DOI 10.1111/cgf.12802
5. Clough, R., Tocher, J.: Finite elements stiffness matrices for analysis of plate bending. In: Proceedings of the 1st Conference on Matrix Methods in Structural Mechanics, vol. 66–80, pp. 515–545. Wright-Patterson A. F. B., Ohio (1965). URL http://contrails.iit.edu/reports/8574
6. Dey, T.K.: Curve and surface reconstruction: algorithms with mathematical analysis. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press (2006). DOI 10.1017/CBO9780511546860
7. Farin, G.: A modified Clough-Tocher interpolant. Comput. Aided Geom. Des. **2**(1–3), 19–27 (1985). DOI 10.1016/0167-8396(85)90003-2
8. Hettinga, G., Kosinka, J.: Multisided generalisations of Gregory patches. Comput. Aided Geom. Des. **62**, 166–180 (2018). DOI 10.1016/j.cagd.2018.03.005
9. Lodha, S., Franke, K.: Scattered data techniques for surfaces. In: Proceedings of Dagstuhl Conference on Scientific Visualization, pp. 182–222. IEEE Computer Society Press, Washington (1997). DOI 10.1109/DAGSTUHL.1997.1423115
10. Mann, S., Loop, C., Lounsbery, M., Meyers, D., Painter, J., DeRose, T., Sloan, K.: A survey of parametric scattered data fitting using triangular interpolants. In: H. Hagen (ed.) Curve and Surface Design, pp. 145–172. SIAM, Philadelphia (1992). DOI 10.1137/1.9781611971651.ch8
11. Nielson, G.: A method for interpolating scattered data based upon a minimum norm network. Math. Comput. **40**, 253–271 (1983). DOI 10.1090/S0025-5718-1983-0679444-7
12. Percell, P.: On cubic and quartic Clough-Tocher finite elements. SIAM J. Numer. Anal. **13**(1), 100–103 (1976). DOI 10.1137/0713011
13. Shirman, L., Séquin, C.: Local surface interpolation with Bézier patches. Comput. Aided Geom. Des. **4**(4), 279–295 (1987). DOI 10.1016/0167-8396(87)90003-3
14. Shirman, L., Séquin, C.: Local surface interpolation with shape parameters between adjoining Gregory patches. Comput. Aided Geom. Des. **7**(5), 375–388 (1990). DOI 10.1016/0167-8396(90)90001-8
15. Shirman, L., Séquin, C.: Local surface interpolation with Bézier patches: errata and improvements. Comput. Aided Geom. Des. **8**(3), 217–221 (1991). DOI 10.1016/0167-8396(91)90005-V
16. Sulzer, R., Landrieu, L., Marlet, R., Vallet, B.: A survey and benchmark of automatic surface reconstruction from point clouds (2023). DOI 10.48550/arXiv.2301.13656
17. Vlachkova, K.: An improved algorithm for scattered data interpolation using quartic triangular Bézier surfaces. In: V.A. Garanzha, L. Kamenski, H. Si (eds.) Numerical Geometry, Grid Generation and Scientific Computing, *LNCSE*, vol. 143, pp. 327–339. Springer Nature Switzerland (2021). DOI 10.1007/978-3-030-76798-3_21
18. Vlachkova, K., Radev, K.: Interpolation of data in $\mathbb{R}^3$ using quartic triangular Bézier surfaces. In: AIP Conf. Proc., vol. 2325 (2021). DOI 10.1063/5.0040457
19. You, C., Lim, S., Lim, S., Tan, J., Lee, C.., Khaw, Y.: A survey on surface reconstruction techniques for structured and unstructured data. In: 2020 IEEE Conference on Open Systems (ICOS), pp. 37–42 (2020). DOI 10.1109/ICOS50156.2020.9293685