# Interpolation of Data in $\mathbb{R}^3$ using Quartic Triangular Bézier Surfaces

Krassimira Vlachkova[1,a)] and Krum Radev[1,b)]

[1]*Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", 1164 Sofia, Bulgaria*

[a)]Corresponding author: krassivl@fmi.uni-sofia.bg
[b)]kvradev@uni-sofia.bg

**Abstract.** We consider the problem of interpolation of data in $\mathbb{R}^3$ and propose a solution based on Nielson's minimum norm network and triangular Bézier patches. Our algorithm uses splitting and constructs $G^1$-continuous bivariate interpolant consisting of quartic triangular Bézier surfaces. The algorithm is computationally simple and produces visually pleasant smooth surfaces. We have created a program packages for implementation, 3D visualization and comparison of our algorithm and the known Shirman and Séquin's method which is also based on splitting and quartic triangular Bézier patches. The results of our numerical experiments are presented and analysed.

**Keywords:** Scattered data interpolation, Minimum norm networks, Bézier surfaces
**2010 Mathematics Subject Classification:** 65D05, 65D17, 68U05

## Introduction

Interpolation of data points in $\mathbb{R}^3$ by smooth surface is an important problem in applied mathematics which finds applications in various areas such as medicine, architecture, archeology, computer graphics, bioinformatics, scientific visualization, etc. In general the problem can be formulated as follows: 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 containing points $V_i = (x_i, y_i)$, such that $F$ possesses continuous partial derivatives up to a given order, and $F(x_i, y_i) = z_i$.

Various methods for solving this problem were proposed and applied, see e.g. [1–4]. A standard approach to solve the problem consists of two steps [1]:

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.

Shirman and Séquin [5, 6] construct a smooth surface consisting of quartic triangular Bézier surfaces (TBS). Their method assumes that in addition we are given as input the normal vectors at points $P_i$, $i = 1, \ldots, n$. First, Shirman and Séquin construct a smooth cubic curve network defined on the edges of $T$ and degree elevate it to quartic. In this way they increase the number of degrees of freedom required for the smooth connection of adjacent Bézier patches. Next, for every triangle in $T$ Shirman and Séquin apply procedure called *splitting* in which for each triangle in $T$ a macro-patch consisting of three Bézier sub-patches is constructed. Splitting was originally proposed by Clough and Tocher [7] and further developed by Percell [8] and Farin [9] for solving different problems. To compute the inner Bézier control points that are close to the edges of $T$, Shirman and Séquin use a method proposed by Chiyokura and Kimura [10, 11], which was originally proposed in a different setting. The resulting interpolation surfaces often suffer from unwanted bulges, tilts, and shears as pointed out by the authors in [12] and more recently by Hettinga and Kosinka in [13].

Nielson [14] proposes a method called *minimum norm network* (MNN) which computes a smooth interpolation cubic curve network defined on the edges of $T$ so as to satisfy an extremal property. A significant advantage of Nielson's method is that simultaneously with the MNN, the normal vectors at the data points are computed. Next, MNN is extended to a smooth interpolation surface using an appropriate *blending* method based on convex combination schemes. Nielson's interpolant is a rational function on every triangle in $T$ and consecutively may have large values in terms of energy.

Here we present a new algorithm for interpolation of data in $\mathbb{R}^3$ which is computationally simple and produces visually pleasant smooth surfaces. Our goal is to construct a smooth interpolation surface that is piecewise polynomial and consists of patches of smallest possible degree. The algorithm first computes the MNN and degree elevate it to quartic, and then constructs a smooth interpolation surface which consists of quartic TBS and is based on splitting. The main differences with Shirman and Séquin's method are as follows. Our choice of the inner Bézier control points allows to avoid possible distortions and twists which appear in surfaces constructed by Shirman and Séquin's method [5, 6]. Hence, this choice improve the quality of the resulting surfaces. Furthermore, Shirman and Séquin impose an additional condition that the three quartic curves defined on the common edges of the three sub-patches are degree elevated cubic curves. This condition is not necessary to obtain $G^1$-continuity across the common edges of the sub-patches. We use different condition for the inner points of the sub-patches and believe that such a choice would facilitate the construction of convex macro-patches.

We have created a program package for implementation, 3D visualization and comparison of Shirman and Séquin's algorithm and the new algorithm. We have chosen `Plotly` [15] and `Three.js` [**?** ] graphics libraries as our main implementation and visualization tools. We performed a large number of experiments using data of increasing complexity and analysed the results with respect to different criteria.

The paper is organized as follows.

## Methods...

In this section we propose an algorithm to construct $G^1$-continuous interpolation surface consisting of quartic TBP.

We first compute the MNN which

## Splitting

For each triangle in $T$ the splitting procedure constructs three triangular Bézier patches (sub-patches) that possess a common vertex and form a $G^1$-continuous polynomial surface (macro-patch) defined in the triangle. The Bézier patches are computed through computation of their control points. Splitting adds more degrees of freedom which allows to connect smoothly the neighbouring macro-patches.

We compute the control points of the three Bézier sub-patches consecutively in four layers as shown in Fig. 1. The first layer consists of inner control points that are nearest to the boundary of $\tau$. The last fourth layer contains of single point $\mathbf{z}$. This point $\mathbf{z}$ is the splitting point and will be computed as a center of the triangle with vertices in the previous third layer. We use the following notation.

- ■ vertices of the sub-patches;
- ● inner control points on the boundary of the macro-patch;
- □ inner control points of the three inner boundary curves of the sub-patches;
- ○ inner control points of the sub-patches.

*Computing the control points in the first layer*. There are three points of type □ and six points of type ○ in this layer (see Fig. 1b). First, we compute points of type □ as centers of the three small triangles with vertices ●■● on the boundary of the macro-patch. Let $\tau$ be the last triangle in a group of triangles around $\mathbf{v}$, see Fig. 1**a.** where the control points $\mathbf{q}_i$, $i = 1, 2$ of the corresponding cubic curve are denoted by •. We compute the points $\mathbf{z}_i$, $i = 1, 2$, of type ○ as follows.

$$\begin{aligned} \mathbf{z}'_i &= \hat{\mathbf{q}}^1_1 - \hat{\mathbf{q}}^1_0 + \mathbf{q}_i, \\ \mathbf{z}''_i &= \hat{\mathbf{q}}^2_1 - \hat{\mathbf{q}}^2_0 + \mathbf{q}_i, \\ \tilde{\mathbf{z}}_i &= (1 - \frac{i}{3})\mathbf{z}'_i + \frac{i}{3}\mathbf{z}''_i, \ i = 1, 2. \end{aligned} \tag{1}$$

The first two coordinates of $\mathbf{z}_i$, $i = 1, 2$, are the same as of $\tilde{\mathbf{z}}_i$, $i = 1, 2$, respectively. We note that the projections of $\mathbf{z}_i$, $i = 1, 2$, onto $Oxy$ lie inside $\tau$. In this way we avoid unwanted twisting and tilting of the patch. The third coordinates of $\mathbf{z}_i$, $i = 1, 2$, are computed as shown in subsection **??**.

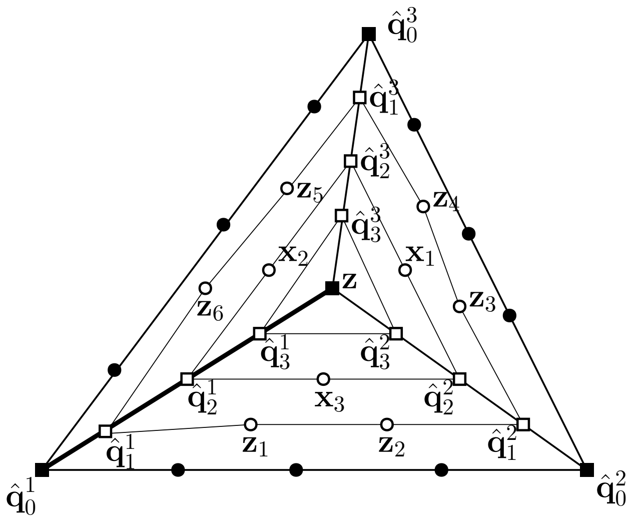

**FIGURE 1.** Construction of the $G^1$-continuous Bézier patch by splitting: **a.** First layer: control points $\mathbf{z}_i$, $i = 1, 2$, are computed by solving the vertex enclosure problem and a minimization procedure; **b.** The 19 control points of the three sub-patches are computed using Algorithm 2 for splitting.

*Computing the control points in the second and third layers.* Now we already know the control points of type ■ and type ● on the boundary of the macro-patch and control points □,○ in the first layer. Let us consider an inner boundary curve, see Fig. 1b and the curve with control points $\hat{\mathbf{q}}_i^1, \mathbf{z}$, $i = 0, \ldots, 3$. We have to compute the remaining control points so as to satisfy the $G^1$-continuity conditions (**??**) across that curve. Points $\hat{\mathbf{q}}_1^1$ and $\mathbf{z}$ are centers of the corresponding small triangle ●■● and triangle $\hat{\mathbf{q}}_3^1\hat{\mathbf{q}}_3^2\hat{\mathbf{q}}_3^3$ where points $\hat{\mathbf{q}}_3^1, \hat{\mathbf{q}}_3^2, \hat{\mathbf{q}}_3^3$ are unknown for now. We have $\alpha_0 = \alpha_1 = 1/2$ and $\beta_0 = \beta_1 = -1/2$. The first equation of system (**??**) is

$$\frac{\mathbf{z}_1 + \mathbf{z}_6}{2} = \mathbf{s}_1 = -\frac{1}{2}\hat{\mathbf{q}}_1^1 + \frac{3}{2}\hat{\mathbf{q}}_2^1 \Rightarrow \hat{\mathbf{q}}_2^1 = \frac{1}{3}(\hat{\mathbf{q}}_1^1 + \mathbf{z}_1 + \mathbf{z}_6).$$

Hence $\hat{\mathbf{q}}_2^1$ is a center of the triangle $\mathbf{z}_1\hat{\mathbf{q}}_1^1\mathbf{z}_6$. Analogously, for the third equation of (**??**) we have

$$\frac{\mathbf{x}_3 + \mathbf{x}_2}{2} = \mathbf{s}_3 = -\frac{1}{2}\hat{\mathbf{q}}_2^1 + \frac{3}{2}\hat{\mathbf{q}}_3^1 \Rightarrow \hat{\mathbf{q}}_3^1 = \frac{1}{3}(\hat{\mathbf{q}}_2^1 + \mathbf{x}_2 + \mathbf{x}_3)$$

and hence $\hat{\mathbf{q}}_3^1$ is a center of triangle $\mathbf{x}_3\hat{\mathbf{q}}_2^1\mathbf{x}_2$. We note that it is not necessary for the inner quartic Bézier curves to be degree elevated cubic curves since we have $\alpha_0 = \alpha_1$, $\beta_0 = \beta_1$ and the second equation (**??**) of system (**??**) is automatically satisfied.

Suppose that we know points $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$ in the second layer. Then points $\hat{\mathbf{q}}_3^1$,$\hat{\mathbf{q}}_3^2$,$\hat{\mathbf{q}}_3^3$ in the third layer are

$$\hat{\mathbf{q}}_3^1 = (\hat{\mathbf{q}}_2^1 + \mathbf{x}_2 + \mathbf{x}_3)/3,\ \hat{\mathbf{q}}_3^2 = (\hat{\mathbf{q}}_2^2 + \mathbf{x}_1 + \mathbf{x}_3)/3,\ \hat{\mathbf{q}}_3^3 = (\hat{\mathbf{q}}_2^3 + \mathbf{x}_1 + \mathbf{x}_2)/3. \quad (2)$$

Finally, the splitting point is $\mathbf{z} := \frac{1}{3}(\hat{\mathbf{q}}_3^1 + \hat{\mathbf{q}}_3^2 + \hat{\mathbf{q}}_3^3)$ and the vertex enclosure problem is satisfied automatically.

It remains to compute the three points $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$. They provide three degrees of freedom which can be used for example to control the shape of the surface. In [5, 6] Shirman and Séquin use the condition that the three quartic curves defined on the inner edges are degree elevated cubic curves as it is for the boundary curves of the macro-patch. As we have already pointed out, this condition is not necessary for $G^1$-continuity between two sub-patches and then we do not need to impose it. We compute points $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$ as mid-points of the edges of the triangle $\hat{\mathbf{q}}_2^1\hat{\mathbf{q}}_2^2\hat{\mathbf{q}}_2^3$ as follows.

$$\mathbf{x}_1 = (\hat{\mathbf{q}}_2^2 + \hat{\mathbf{q}}_2^3)/2,\ \mathbf{x}_2 = (\hat{\mathbf{q}}_2^1 + \hat{\mathbf{q}}_2^3)/2,\ \mathbf{x}_3 = (\hat{\mathbf{q}}_2^1 + \hat{\mathbf{q}}_2^2)/2.$$

Then the control points in second, third, and fourth layers become co-planar. We believe that this choice of $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$ will be useful when we try to construct a simple macro-patch which is convex in addition.

Algorithm 1 below takes a triangle $\tau$ in $T_s$ and the degree-elevated quartic boundary control points of the corresponding patch and computes 19 control points of the three $G^1$-continuous quartic Bézier sub-patches.

**Algorithm 1** Splitting

```
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 subsection ??.
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.
```

## Examples

We implemented our Algorithm **??** as a web application using HTML, JavaScript, and the open source library `Plotly` [15]. The advantages of using `Plotly` are the options to display the coordinate system and the coordinates of the points when the cursor is placed on them, and to control the surfaces displayed. To demonstrate the results of our work we present here the following example that clearly show the main differences between Shirman and Séquin's and our methods.

**Example 1** We consider data obtained from a regular triangular pyramid. We have $N = 4$, $\mathbf{v}_1 = (-1/2, -\sqrt{3}/6)$, $\mathbf{v}_2 = (1/2, -\sqrt{3}/6)$, $\mathbf{v}_3 = (0, \sqrt{3}/3)$, $\mathbf{v}_4 = (0, 0)$, and $z_i = 0$, $i = 1, 2, 3$, $z_4 = -1$. The triangulation and the corresponding MNN are shown in Fig. 3. In this case $T_s$ is empty and no triangle has been splitted. The corresponding Shirman and Séquin's surface is shown in Fig. 3. The surface generated by our Algorithms **??** and 1 is shown in Fig. **??**.

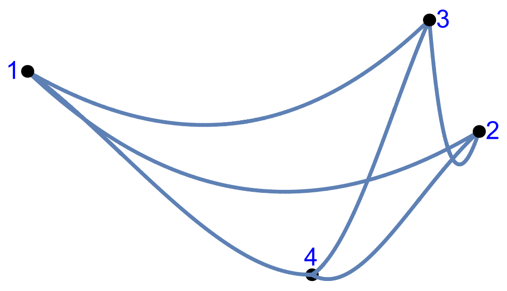

**FIGURE 2.** The MNN for the data in Example 1

## Conclusion and Future Work

In this paper we have proposed a method for constructing a $G^1$-continuous interpolation surface that consists of triangular quartic Bèzier patches. We note that while splitting decreases the smoothness of a polynomial macro-patch to $G^1$ only, on the other hand, it is not necessary to apply splitting to all triangles in $T$. Hence, a challenging idea is to construct an interpolation surface that is piecewise polynomial and consists of the least number of Bèzier patches of smallest possible degree. Such interpolants are preffered in practice since they are computationally simple and have higher degree of smoothness.

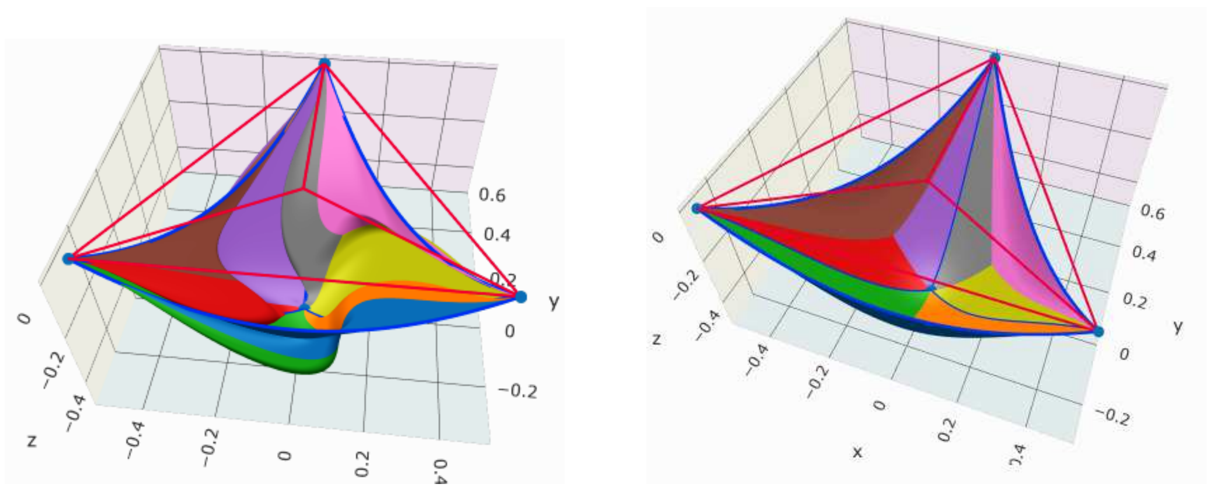

**FIGURE 3.** Shirman and Séquin's surface for the data in Example 1.

## ACKNOWLEDGMENTS

This research was supported by Sofia University Science Fund Grant No. 80-10-171/2020, and by European Regional Development Fund and the Operational Program "Science and Education for Smart Growth" under Contract BG05M2OP001-1.001-0004 (2018-2023).

## REFERENCES

[1] S. Mann, C. Loop, M. Lounsbery, D. Meyers, J. Painter, T. DeRose, and K. Sloan, in *Curve and Surface Design*, edited by H. Hagen (SIAM, Philadelphia, 1992), pp. 145–172.

[2] T. K. Dey, *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*, Cambridge Monographs on Applied and Computational Mathematics (Cambridge University Press, 2006).

[3] K. Anjyo, J. Lewis, and F. Pighin, Scattered data interpolation for computer graphics, SIGGRAPH 2014 course notes, 2014, last accessed June 20, 2020.

[4] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, G. Guennebaud, J. Levine, A. Sharf, and C. Silva, Comput. Graph. Forum **36**, 301 – 329 (2017).

[5] L. Shirman and C. Séquin, Comput. Aided Geom. Des. **4**, 279–295 (1987).

[6] L. Shirman and C. Séquin, Comput. Aided Geom. Des. **8**, 217–221 (1991).

[7] R. Clough and J. Tocher, "Finite elements stiffness matrices for analysis of plate bending," in *Proceedings of the 1st Conference on Matrix Methods in Structural Mechanics*, Vol. 66–80 (Wright-Patterson A. F. B., Ohio, 1965), pp. 515–545.

[8] P. Percell, SIAM J. Numer. Anal. **13**, 100–103 (1976).

[9] G. Farin, Comput. Aided Geom. Des. **2**, 19–27 (1985).

[10] H. Chiyokura and F. Kimura, "Design of solids with free-form surfaces," in *SIGGRAPH '83 Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, Vol. 17, edited by P. P. Tanner (ACM, New York, 1983), pp. 289–298.

[11] H. Chiyokura, "Localized surface interpolation method for irregular meshes," in *Advanced Computer Graphics, Proceedings of Computer Graphics Tokyo'86*, Vol. 66–80, edited by T. Kunii (Springer, Tokyo, 1986), pp. 3–19.

[12] L. Shirman and C. Séquin, Comput. Aided Geom. Des. **7**, 375–388 (1990).

[13] G. Hettinga and J. Kosinka, Comput. Aided Geom. Des. **62**, 166 – 180 (2018).

[14] G. Nielson, Math. Comput. **40**, 253–271 (1983).

[15] Plotly.js, 2020, last accessed June 20.