algorithms

MDPI

Article

Cubic q-Bézier Triangular Patch for Scattered Data Interpolation and Its Algorithm

Owen Tamin [1] and Samsul Ariffin Abdul Karim [2,*]

[1] Faculty of Science and Natural Resources, Universiti Malaysia Sabah, Jalan UMS, Kota Kinabalu 88400, Malaysia; owentamin1996@yahoo.com
[2] School of Quantitative Sciences, UUM College of Arts and Sciences, Universiti Utara Malaysia, Sintok 06010, Malaysia
* Correspondence: samsul.ariffin@uum.edu.my

Abstract: This paper presents an approach to scattered data interpolation using q-Bézier triangular patches via an efficient algorithm. While existing studies have formed q-Bézier triangular patches through convex combination, their application to scattered data interpolation has not been previously explored. Therefore, this study aims to extend the use of q-Bézier triangular patches to scattered data interpolation by achieving C^1 continuity throughout the data points. We test the proposed scheme using both established data points and real-life engineering problems. We compared the performance of the proposed interpolation scheme with a well-known existing scheme by varying the q parameter. The comparison was based on visualization and error analysis. Numerical and graphical results were generated using MATLAB. The findings indicate that the proposed scheme outperforms the existing scheme, demonstrating a higher coefficient of determination (R^2), smaller root mean square error (RMSE), and faster central processing unit (CPU) time. These results highlight the potential of the proposed q-Bézier triangular patches scheme for more accurate and reliable scattered data interpolation via the proposed algorithm.

Keywords: patches; scattered data interpolation; continuity; visualization; surface reconstruction

check for updates

Citation: Tamin, O.; Karim, S.A.A. Cubic q-Bézier Triangular Patch for Scattered Data Interpolation and Its Algorithm. *Algorithms* **2024**, *1*, 422. https://doi.org/10.3390/a17090422

Academic Editors: Shuai Li and Dunhui Xiao

Received: 10 July 2024
Revised: 5 August 2024
Accepted: 7 August 2024
Published: 23 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Scattered data interpolation is crucial in computational science, providing a method to estimate values at arbitrary points within datasets characterized by irregularly distributed data points. Unlike structured datasets, which are arranged on grids or meshes, scattered data often arise from real-world observations or simulations where data points are irregularly positioned in multidimensional space. Interpolating these data involves constructing a continuous function that approximates the dataset's underlying behavior, allowing for the inference of values at unobserved locations within the domain [1].

There are two primary approaches to scattered data interpolation: triangulation-based methods and meshless techniques. Triangulation-based approaches involve creating a mesh from the scattered data points. However, not all interpolation methods require such a mesh. For example, some variants of the Shepard method do not require creating a mesh; instead, barycentric coordinates are used to obtain the explicit expression of the polynomial interpolant [2]. Common basis functions used in these methods include Bézier, B-spline, and radial basis functions (RBFs). To mitigate noise in the scattered data, least squares approximation is often employed.

Liu [3] developed a local multi-level scattered data interpolation method by introducing layered datasets and adjusting compactly supported radial basis functions. Borne and Wende [4,5] explored meshless methods using specific RBFs for interpolation, employing domain decomposition methods to generate symmetric saddle point systems. Skala et al. [5] presented an innovative variant of the normalized RBFs known as the Squared Normalized RBFs (SN-RBFs), specifically designed for spatio-temporal data. This new formulation

involves the normalization of the rows of the RBF matrix, a process that significantly improves the conditioning of the RBF matrix. As a result, the SN-RBF method enhances the robustness and accuracy of RBF interpolation, making it more suitable for a wide range of applications.

Joldes et al. [6] enhanced the moving least squares (MLS) technique by incorporating polynomial bases to improve scattered data interpolation. Brodlie et al. [7] examined constrained surface interpolation using the Shepard interpolant, addressing the problem with optimization techniques. Lai and Meile [8] reviewed the use of non-negative bivariate triangular splines for maintaining the shape of scattered data. Schumaker and Speleers [9] investigated preserving non-negativity using macro-element spline spaces with Clough–Tocher macro-elements.

Karim et al. [10,11] explored spatial interpolation for rainfall data and terrain data using cubic Bézier and cubic Ball triangular patches. These studies demonstrated that the scattered data interpolation approach by Said and Rahmat [12] does not exhibit C^1 continuity, prompting the establishment of new criteria for C^1 continuity. Their method yielded a differentiable surface with a seamless appearance.

Feng and Zhang [13] introduced segmented bivariate Hermite interpolation using triangulation for precise surface reconstruction from extensive scattered datasets. Sun et al. [14] developed a bivariate rational interpolation on a triangular domain but noted increased computation time with rational splines. Bozzini et al. [15] suggested using polyharmonic splines to estimate noisy scattered data.

Goodman and Said [16] developed a C^1 triangular interpolant using a convex combination scheme, differing from Foley and Opitz's C^1 cubic triangular convex combination technique [17]. Said and Rahmat's method using cubic Ball triangular patches yielded results equivalent to cubic Bézier triangular patches with reduced computation [12]. Karim and Saaban's study confirmed the non-continuous differentiability of Said and Rahmat's function [18].

Hussain et al. [19] introduced rational cubic Bézier triangular methods for positivity-preserving interpolation, asserting continuous differentiability. Chan and Ong [20] examined range-restricted interpolation using cubic Bézier triangles with partial derivatives calculated using Goodman's approach.

Research has also explored quartic Bézier triangular patches for scattered data interpolation. Saaban et al. [21] and Piah et al. [22] developed C^1 (or G^1) methods using quartic Bézier triangles, optimizing inner Bézier points to maintain range and positivity. Hussain et al. [23] expanded this to convexity-preserving interpolation with rational quartic Bézier triangles, addressing optimization challenges for C^1 continuity. Comprehensive surveys on scattered data interpolation are available in sources like [1,24–27].

Among the various generalizations of Bézier curves, q-Bézier curves introduce greater versatility to the classical Bézier curves through the use of the parameter q. This parameter allows for more flexible and dynamic curve modeling, accommodating a wider range of shapes and applications [28–33]. Lewanowicz et al. [30] expanded the q-Bernstein basis of univariate polynomials to a triangular domain, providing a novel framework for handling scattered data.

However, the evaluation algorithm they proposed does not generally involve linear convex combinations, which limits its practical utility in some applications [30]. Foley [17] addressed this limitation by extending the evaluation algorithm to incorporate convex combinations, enhancing the algorithm's applicability and robustness. Inspired by these advancements, we aim to further explore and apply these extended q-Bézier curve techniques to the challenging problem of scattered data interpolation, leveraging their flexibility to achieve more accurate and efficient interpolation results.

2. q-Bézier Triangular Patch

Definition 1. *Given a non-negative integer n and three indices $i, j, k \geq 0$ such that $i + j + k = n$, the triangular q-Bernstein from [34] is defined as*

$$B^n_{i,j,k}(u,v) = \begin{bmatrix} n \\ k \end{bmatrix} \begin{pmatrix} i+j \\ i \end{pmatrix} u^i v^j \prod_{s=0}^{k-1}(1-q^s u - q^s v), \quad i,j,k \geq 0, \quad i+j+k=n \tag{1}$$
$$\text{where } q \in [0,1]$$

In order to construct a q-Bézier triangular patch for scattered data interpolation, the existing triangular Bernstein polynomials $B^n_{i,j,k}(u,v)$ are extended to $B^n_{i,j,k}(u,v,w)$, which is defined in Proposition 1.

Proposition 1. *The triangular q-Bernstein polynomials from Equation (1) are extended to* $B^n_{i,j,k}(u,v,w)$*, which are given by*

$$B^n_{i,j,k}(u,v,w) = \begin{bmatrix} n \\ k \end{bmatrix} \begin{pmatrix} i+j \\ i \end{pmatrix} u^i v^j \prod_{s=0}^{k-1}(1-q^s + q^s w), \quad i,j,k \geq 0, \quad i+j+k=n \tag{2}$$
$$\text{where } q \in [0,1]$$

Proof. Given a non-negative integer n and three indices $i,j,k \geq 0$ such that $i+j+k=n$,

$$B^n_{i,j,k}(u,v) = \begin{bmatrix} n \\ k \end{bmatrix} \begin{pmatrix} i+j \\ i \end{pmatrix} u^i v^j \prod_{s=0}^{k-1}(1-q^s u - q^s v) \tag{3}$$
$$= \begin{bmatrix} n \\ k \end{bmatrix} \begin{pmatrix} i+j \\ i \end{pmatrix} u^i v^j \prod_{s=0}^{k-1}(1-q^s u - q^s v) + q^s - q^s \tag{4}$$

By factorizing Equation (4), we have

$$B^n_{i,j,k}(u,v) = \begin{bmatrix} n \\ k \end{bmatrix} \begin{pmatrix} i+j \\ i \end{pmatrix} u^i v^j \prod_{s=0}^{k-1}(1-q^s + q^s(1-u-v)) \tag{5}$$

Taking into account that $w = 1-u-v$, we can therefore rewrite $B^n_{i,j,k}(u,v)$ as

$$B^n_{i,j,k}(u,v,w) = \begin{bmatrix} n \\ k \end{bmatrix} \begin{pmatrix} i+j \\ i \end{pmatrix} u^i v^j \prod_{s=0}^{k-1}(1-q^s + q^s w) \tag{6}$$

□

Definition 2. *Let* $D := \{(u,v,w) \mid u+ v+w = 1, u > 0, v > 0, w > 0\}$*. The following ten functions are a cubic q-Bézier basis function over the triangular domain* D *shown in Figure 1.*

Let $\alpha = 1+q+q^2$, $\beta = 1-q+qw$, and $\gamma = 1-q^2+q^2w$.

$$\begin{aligned}
B^3_{3,0,0}(u,v,w) &= u^3,\\
B^3_{0,3,0}(u,v,w) &= v^3,\\
B^3_{0,0,3}(u,v,w) &= \beta\gamma w,\\
B^3_{2,1,0}(u,v,w) &= 3u^2 v,\\
B^3_{2,0,1}(u,v,w) &= \alpha u^2 w,\\
B^3_{1,2,0}(u,v,w) &= 3uv^2,\\
B^3_{0,2,1}(u,v,w) &= \alpha v^2 w,\\
B^3_{1,0,2}(u,v,w) &= \alpha\beta uw,\\
B^3_{0,1,2}(u,v,w) &= \alpha\beta vw,\\
B^3_{1,1,1}(u,v,w) &= 1 - \sum_{i+j+k=3, i,j,k\neq 1} B^3_{i,j,k}(u,v,w)
\end{aligned}$$

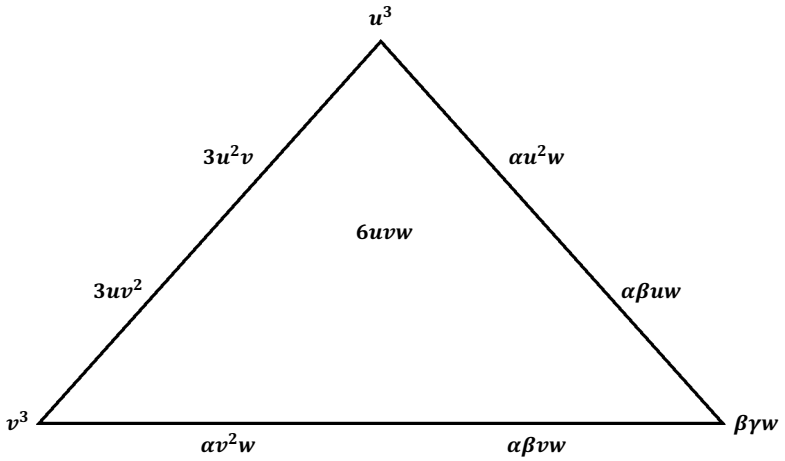


Figure 1. Bases for q-Bézier triangular.

Theorem 1. *The basis function defined over a triangular domain has the following properties:*

(a) *Partition of unity:* $\sum_{i+j+k=3} B^3_{i,j,k}(u,v,w) = 1$.

(b) *Non-negativity:* $B^3_{i,j,k}(u,v,w) \geq 0,\ i,j,k \in \mathbb{N},\ i+j+k=3$.

(c) *Property at the corner points: for* $i,j,k \in \mathbb{N},\ i+j+k=3$, *we have*

$$B^3_{i,j,k}(1,0,0) = \begin{cases} 1, & i=3, \\ 0, & i \neq 3, \end{cases}$$

$$B^3_{i,j,k}(0,1,0) = \begin{cases} 1, & j=3, \\ 0, & j \neq 3, \end{cases}$$

$$B^3_{i,j,k}(0,0,1) = \begin{cases} 1, & k=3, \\ 0, & k \neq 3, \end{cases}$$

(d) *When* $q=1$, *the cubic q-Bézier triangular is reduced to the standard cubic Bézier.*

Definition 3. *A cubic q-Bézier triangular patch is defined as*

$$R(u,v,w) = \sum_{i+j+k=3} B^3_{i,j,k}(u,v,w) b_{i,j,k} \tag{7}$$

where $B^3_{i,j,k}(u,v,w)$ *is defined by Equation (6) for all* $q \in [0,1]$.

Figure 2 presents a comparison of the interpolation surfaces for one cubic q-Bézier patch, with varying values of q where the values are 0.25, 0.5, 0.75, and 1.0. As the value of q increases, the surface gradually shifts closer to the control polygon, reflecting the influence of q on the overall interpolation. When $q = 1.0$, the surface conforms more closely to the shape of the control polygon, while for smaller q values (e.g., 0.25), the surface tends to be smoother and less influenced by the control points. This demonstrates the flexibility of the q-Bézier interpolation scheme in controlling the tension and shape of the interpolating surface.

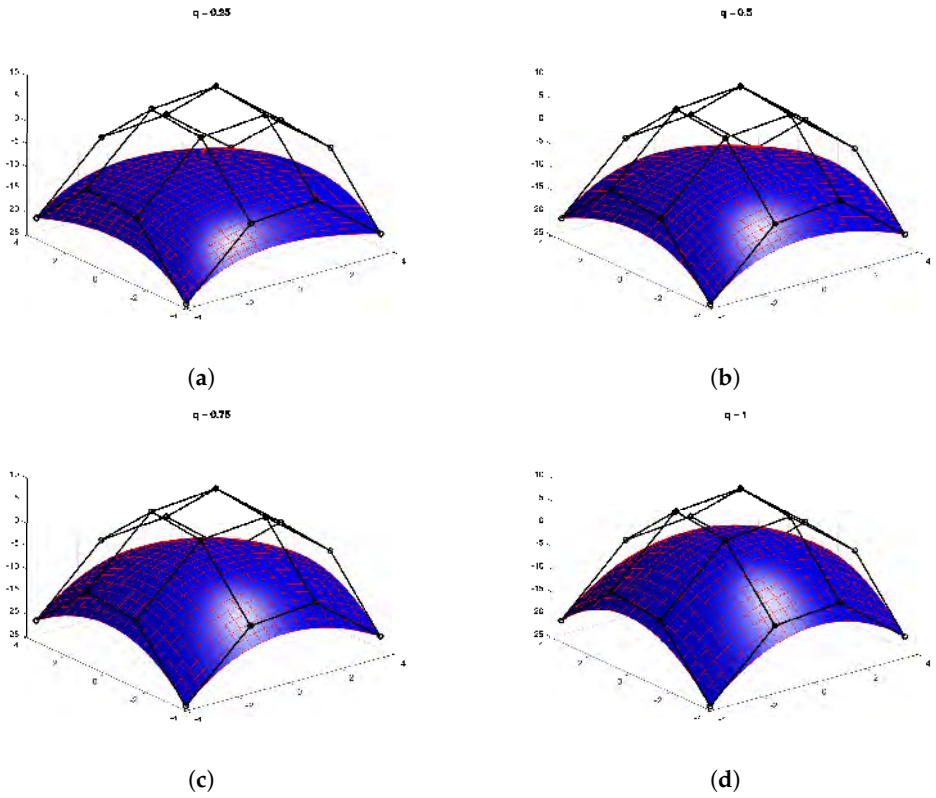


Figure 2. Cubic q-Bézier triangular with different values of q.

3. Scattered Data Interpolation Using q-Bézier Triangular Patch

In this section, we will extend the cubic q-Bézier triangular patch discussed in Section 2 to the application of scattered data interpolation. Figure 3 shows the arrangement of control points on one patch of a cubic q-Bézier triangular patch.

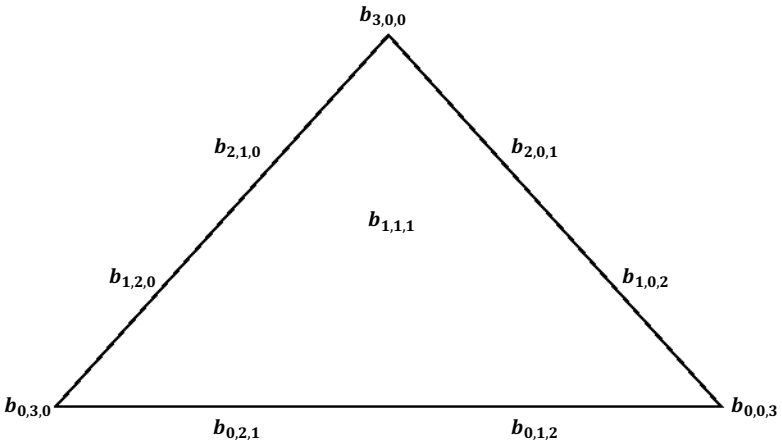


Figure 3. Control points for cubic q-Bézier triangular patch.

3.1. Derivative Estimation

The partial derivatives $f_x(V_i)$ and $f_y(V_i)$ at the vertices $(x_i, y_i), i = 1, 2, 3, \ldots, n$ of th triangle are estimated using the method proposed in [16]. These partial derivatives are used to find derivatives in the direction of ε, which is an edge of the triangle that is shown in Figure 4.

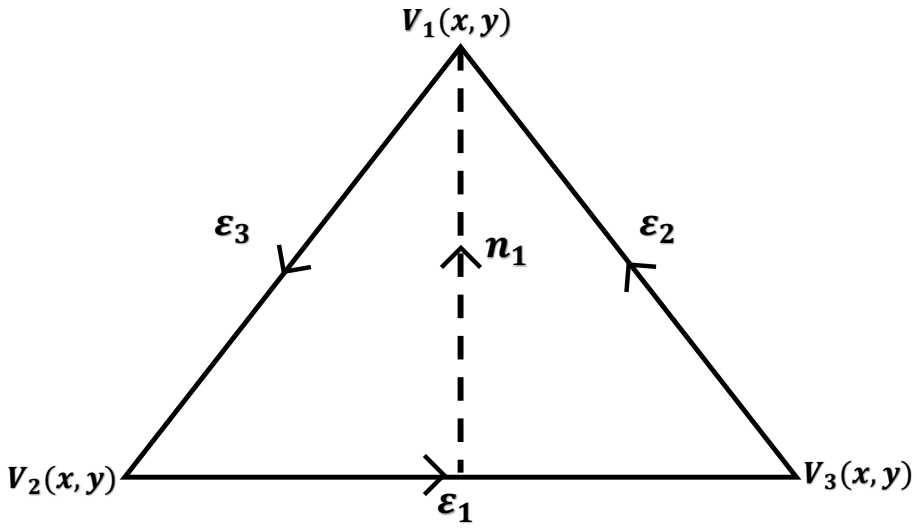


Figure 4. Directionals ε_1, ε_2, and ε_3.

The derivative along the side ε_i is given by

$$\frac{\partial f}{\partial \varepsilon_i} = (x_{i-1} - x_{i+1})\frac{\partial f}{\partial x} + (y_{i-1} - y_{i+1})\frac{\partial f}{\partial y} \tag{8}$$

3.2. Boundary Bézier Ordinates

Boundary Bézier ordinates are calculated using the C^1 continuity condition at the vertices of the triangle. At each vertex there are two directional derivatives in the direction of both the edges incident at the vertex.

The q-Bézier triangular patch is C^1 at the vertices of the triangle if the boundary Bézier ordinates are

$$b_{3,0,0} = F(V_1)$$
$$b_{2,0,1} = \frac{(q^3-1)b_{1,0,2} + (-q^3+q^2+q-1)F(V_3) + 3F(V_1)}{q^2+q+1} - \frac{1}{q^2+q+1}\frac{\partial F(V_1)}{\partial e_2}$$
$$b_{2,1,0} = F(V_1) + \frac{1}{3}\frac{\partial F(V_1)}{\partial e_3}$$
$$b_{0,3,0} = F(V_2)$$
$$b_{0,2,1} = \frac{(q^3-1)b_{0,1,2} + (-q^3+q^2+q-1)F(V_3) + 3F(V_2)}{q^2+q+1} + \frac{1}{q^2+q+1}\frac{\partial F(V_2)}{\partial e_1}$$
$$b_{1,2,0} = F(V_2) - \frac{1}{3}\frac{\partial F(V_2)}{\partial e_3}$$
$$b_{0,0,3} = F(V_3)$$
$$b_{0,1,2} = F(V_3) - \frac{1}{q^2+q+1}\frac{\partial F(V_3)}{\partial e_1}$$
$$b_{1,0,2} = F(V_3) + \frac{1}{q^2+q+1}\frac{\partial F(V_3)}{\partial e_2}$$

3.3. Inner Bézier Ordinates

The remaining $b^i_{1,1,1}, i = 1, 2, 3$ is obtained using the cubic precision method of Foley and Opitz [17], as shown in Figure 5. For complete derivation, the reader can refer to [17].

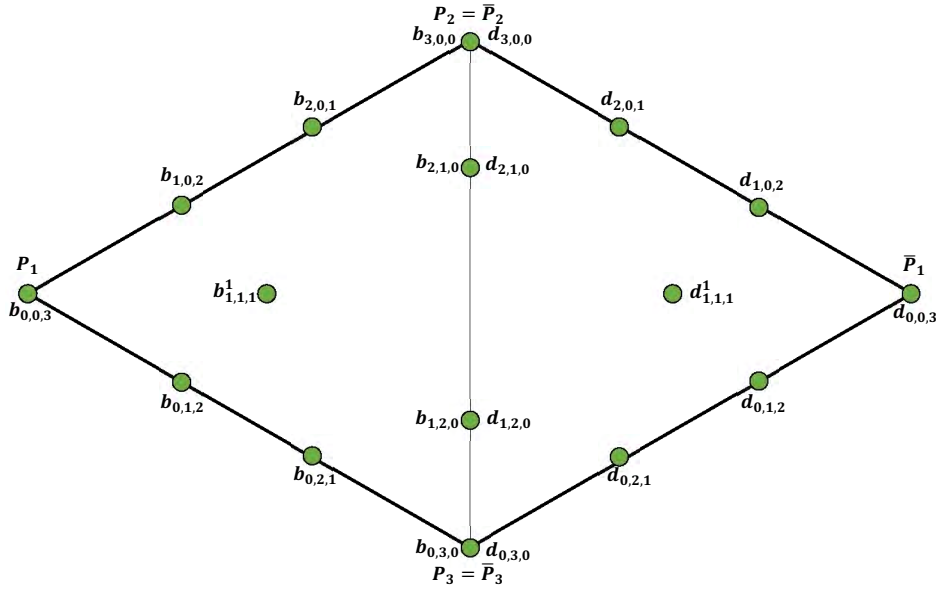


Figure 5. Two adjacent q-Bézier triangular patches.

Let (r, s, t) be the barycentric coordinates of $\bar{P}_1$ with respect to the triangle $P_1P_2P_3$. In order to achieve C^1 continuity along all edges, the following equations must be satisfied:

$$d_{0,1,2} = r^2 b_{0,3,0} + 2rsb_{1,2,0} + 2rtb_{0,2,1} + s^2 b_{2,1,0} + 2stb^1_{1,1,1} + t^2 b_{0,1,2} \tag{9}$$

$$d_{1,0,2} = r^2 b_{1,2,0} + 2rsb_{2,1,0} + 2stb_{2,0,1} + s^2 b_{3,0,0} + 2rtb^1_{1,1,1} + t^2 b_{1,0,2} \tag{10}$$

In reversing the triangles where (u, v, w) are the barycentric coordinates of P_1 with respect to the triangle $\bar{P}_1\bar{P}_2\bar{P}_3$, the following equation is obtained:

$$b_{0,1,2} = u^2 d_{0,3,0} + 2uvd_{1,2,0} + 2uwd_{0,2,1} + v^2 d_{2,1,0} + 2vwd^1_{1,1,1} + w^2 d_{0,1,2} \tag{11}$$

$$b_{1,0,2} = u^2 d_{1,2,0} + 2uvd_{2,1,0} + 2vwd_{2,0,1} + v^2 d_{3,0,0} + 2uwd^1_{1,1,1} + w^2 d_{1,0,2} \tag{12}$$

The value of d^1_{111} is obtained by adding Equations (10) and (11) together. Thus, we obtain

$$\begin{aligned} d^1_{1,1,1} = \frac{1}{2w(u+v)}\big(&-u^2 d_{0,3,0} - u^2 d_{1,2,0} - 2uvd_{1,2,0} - 2uvd_{2,1,0} - 2uwd_{0,2,1} \\ &- v^2 d_{2,1,0} - v^2 d_{3,0,0} - 2vwd_{2,0,1} - w^2 d_{0,1,2} - w^2 d_{1,0,2}\big) \end{aligned} \tag{13}$$

Similarly, by adding Equations (8) and (9), we obtain

$$\begin{aligned} b^1_{1,1,1} = \frac{1}{2t(r+s)}(&-r^2 b_{0,3,0} - r^2 b_{1,2,0} - 2rsb_{1,2,0} - 2rsb_{2,1,0} - 2rtb_{0,2,1} \\ &- s^2 b_{2,1,0} - s^2 b_{3,0,0} - 2stb_{2,0,1} - t^2 b_{0,1,2} - t^2 b_{1,0,2}) \end{aligned} \tag{14}$$

The final scattered data interpolation scheme using the cubic q-Bézier triangular patch can be expressed as follows:

$$R(u, v, w) = \sum_{i+j+k=3} B^3_{i,j,k}(u, v, w) b_{i,j,k} + 6uvw(\tau_1 b^1_{1,1,1} + \tau_2 b^2_{1,1,1} + \tau_3 b^3_{1,1,1}) \tag{15}$$

where

$$\tau_1 = \frac{vw}{uv + vw + wu} \quad , \quad \tau_2 = \frac{wu}{uv + vw + wu} \quad , \quad \tau_3 = \frac{uv}{uv + vw + wu}$$

The computational complexity of our proposed scattered data interpolation scheme for a single triangular patch involves 38 multiplications, 13 additions, and 1 division. This complexity is comparable to that of classical methods, such as those in [16], which require a total of 52 operations.

The overall algorithm used in this study with descriptions is below.

4. Results and Discussion

In this study, we implemented the proposed scheme using MATLAB 2023a to conduct the experiment. The MATLAB coding was developed based on Algorithm 1, and the processor used was 1.6 GHz Dual-Core Intel Core i5 (Intel, Santa Clara, CA, USA). The error norms were computed using a 33×33 uniform rectangular grid of evaluation points in the unit square. The error measurements used were the (a) coefficient of determination, R^2; (b) root mean square error (RMSE); and (c) central processing unit (CPU) time.

Algorithm 1 q-Bézier Triangular Patches for Scattered Data Interpolation

Input: Scattered data points $\{(x_i, y_i, z_i)\}_{i=1}^{n}$
Output: Interpolated surface, R^2, RMSE, and CPU time

Step 1: Estimate Partial Derivatives

- Compute the partial derivatives $\frac{\partial f}{\partial \varepsilon_1}$, $\frac{\partial f}{\partial \varepsilon_2}$ and $\frac{\partial f}{\partial \varepsilon_3}$ at each data point using Equation (8).
- MATLAB functions: `diff`, `gradient`

Step 2: Perform Delaunay Triangulation

- Use a Delaunay triangulation algorithm to create a mesh from the scattered data points.
- MATLAB functions: `delaunay`, `triplot`

Step 3: Calculate Boundary Control Points

- Determine the boundary control points for each triangle in the triangulated mesh.
- MATLAB functions: `convhull`, `boundary`

Step 4: Calculate Inner Control Points

- For each triangle, compute the inner control points for the local scheme $b^i_{1,1,1}, i = 1, 2, 3$.
- MATLAB function: custom implementation based on Foley and Opitz [17]

Step 5: Construct the Interpolated Surface

- Use the convex combination method to combine the three local schemes defined by Equation (15).
- MATLAB functions: `sum`, `minus`, `divide`

Step 6: Evaluate Performance Metrics

- Calculate R^2, RMSE, and CPU time.
- MATLAB functions: `sum`, `abs`, `sqrt`, `mean`, `tic`, `toc`

The error bound for the proposed scattered data interpolation scheme was adopted from [35]. Let the proposed scheme be denoted as p and Φ be the convex hull of points $\{(x_i, y_i)\}, i = 1, 2, \ldots, N$. Let Δ be a triangulation of Φ.

Theorem 2. *For every $f \epsilon C^4(\Phi)$. Its C^1 q-Bézier interpolant p satisfies*

$$\|f - p\| \leq K|\Delta|^4 |f|_4$$

where the constant K depends on the smallest angle in Δ. Furthermore, $|\Delta|$ is the size of the triangulation, i.e., the Δ largest length of edges of Δ.

The proposed scheme was tested using three well-known test functions $F_1(x, y)$, $F_2(x, y)$, and $F_3(x, y)$ for surface interpolation (Karim et al. [36]):

1. Sphere function:

$$F_1(x,y) = 0.75e^{\left(-\frac{(9x-2)^2+(9y-2)^2}{4}\right)} + 0.75e^{\left(-\frac{(9x+1)^2}{49}-\frac{(9y+1)^2}{10}\right)} + 0.5e^{\left(-\frac{(9x-7)^2+(9y-3)^2}{4}\right)} - 0.2e^{\left(-(9x-4)^2-(9y-7)^2\right)}$$

2. Cliff function:

$$F_2(x,y) = \frac{(1.25 + \cos(5.4y))}{6 + 6(3x-1)^2}$$

3. Gentle function:

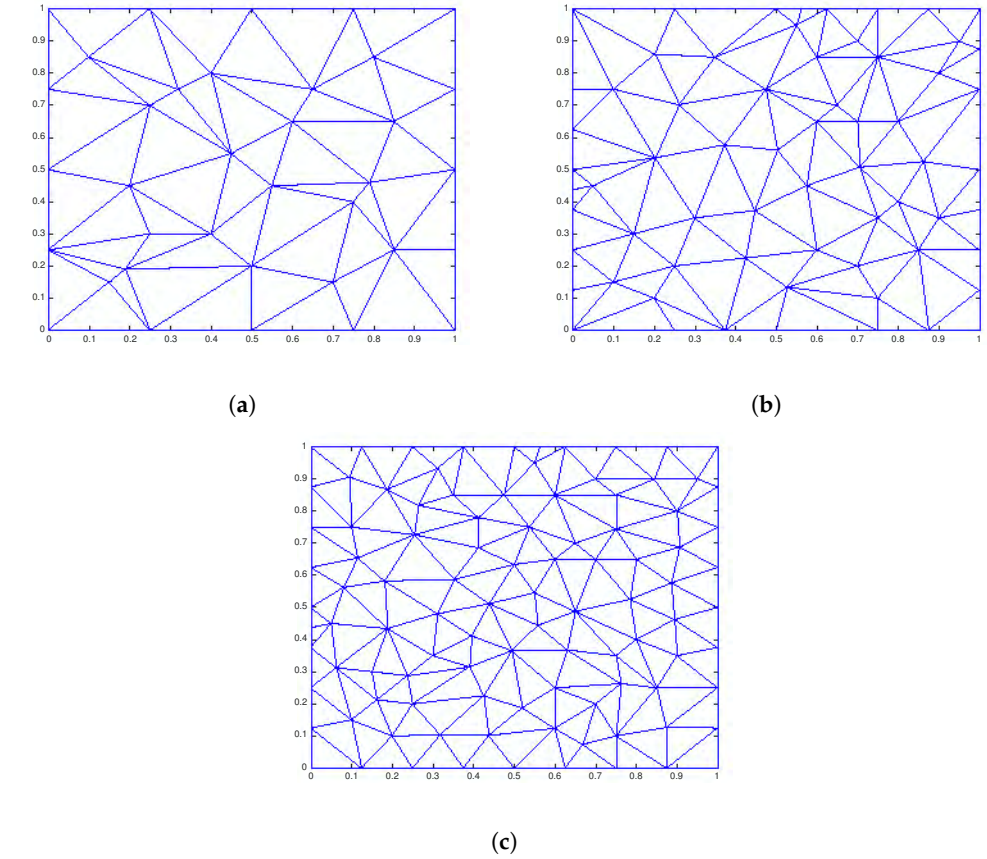
$$F_3(x,y) = \frac{1}{3}e^{-\frac{81}{16}((x-0.5)^2-(y-0.5)^2)}$$

Figure 6 shows the Delaunay triangulation of 36, 65, and 100 data points on the rectangular domain of [0, 1] × [0, 1] with uniform distribution. The Delaunay triangulation presented in Figure 6 is constrained, ensuring that specific segments are included as edges. This approach is particularly useful for maintaining specific boundary conditions and incorporating predefined features into the mesh structure [37].

Figures 7–14 display the surface interpolation and corresponding contour plots with various q parameters and data points derived from the $F_1(x, y)$ function. The visual inspection reveals that surface interpolation and its contour plots exhibit noticeable smoothness with 100 data points compared to 36 and 65 data points. Despite the change in the q parameter, differentiating the resulting surfaces proves visually challenging.

Similarly, Figures 15–22 illustrate surface interpolation and contour plots based on the $F_2(x, y)$ function under a varying q parameter and data points. Consistent with the observations from the $F_1(x, y)$ function, the surface interpolation and its contour plots are visually smoother with 100 data points compared to 36 and 65 data points. However, like before, distinguishing the effects of different q values remains difficult.

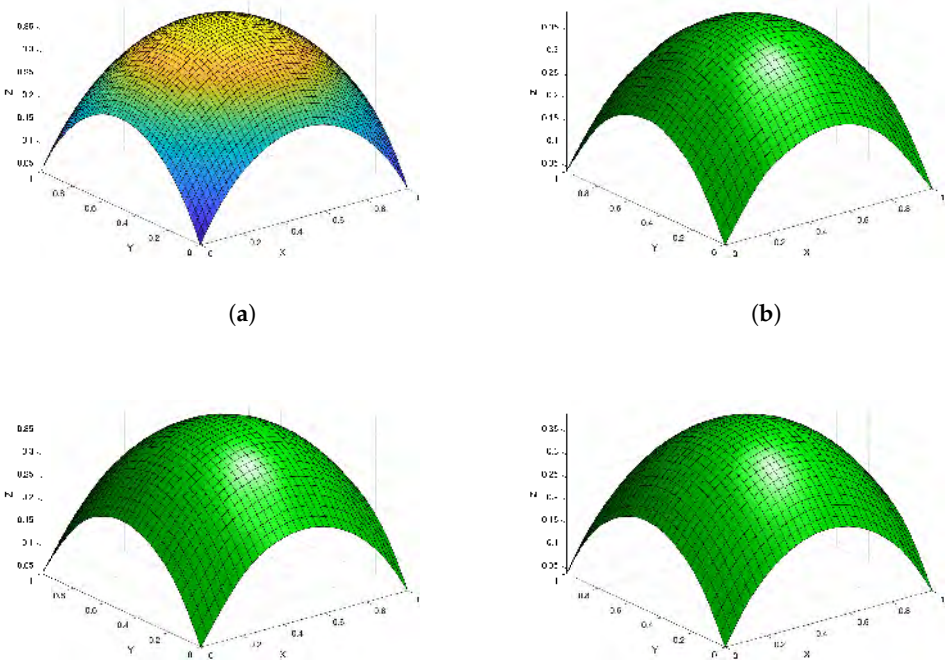
Figures 23–30 present the surface interpolation and contour plots for the $F_3(x, y)$ function, again with different q values and data points. The visual smoothness of surface interpolation for 100 data points, as opposed to 36 and 65 data points, mirrors the trends observed with the $F_1(x, y)$ and $F_2(x, y)$ functions. Across all three test functions, the interpolation results generated with 100 data points are accurate, suggesting that an increased number of data points enhances the interpolation accuracy relative to the true surface. However, the similarity in results across different q parameters indicates that visual inspection alone is insufficient to determine the optimal q parameter value.



(**a**) (**b**)

(**c**)

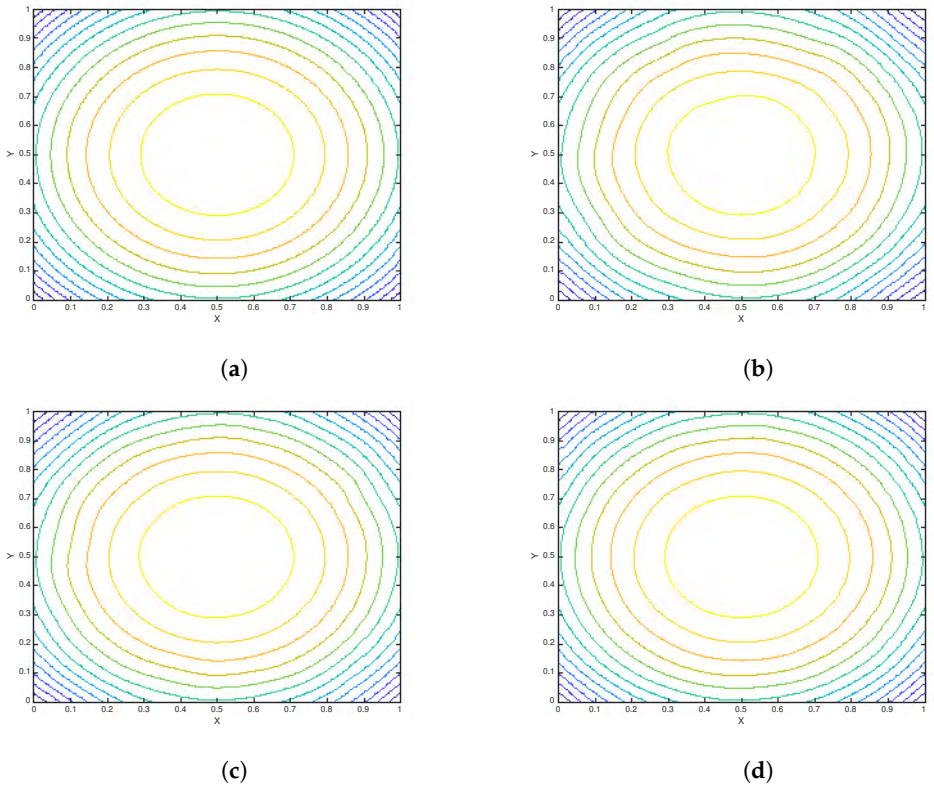
Figure 6. Triangulation domain for (**a**) 36 data points, (**b**) 65 data points, and (**c**) 100 data points.



(**a**) (**b**)

(**c**) (**d**)

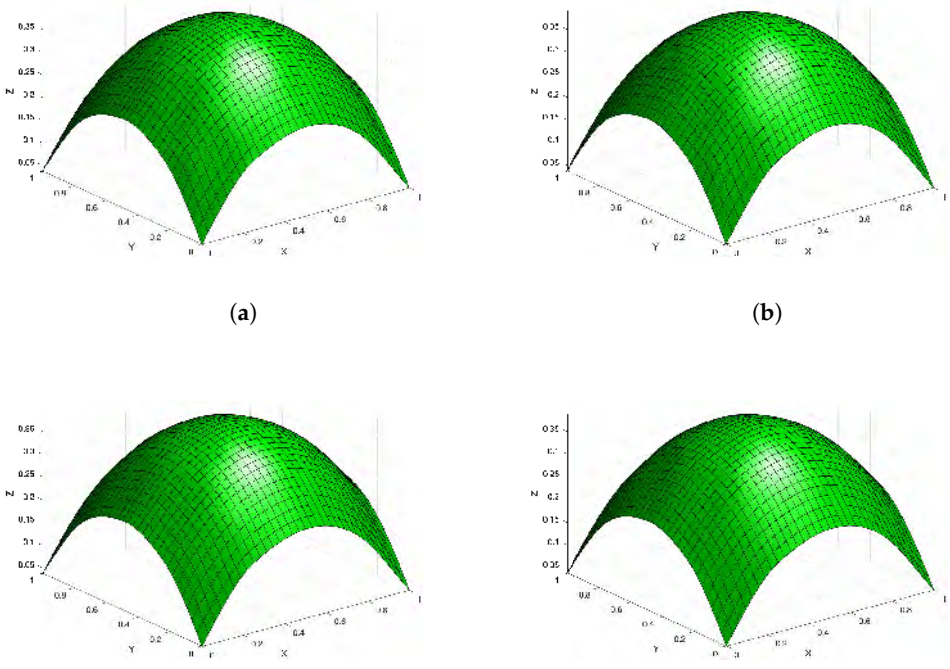
Figure 7. Surface interpolation based on F_1 function with 36, 65, and 100 data points when $q = 1$: (**a**) true surface; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.



(**a**) (**b**)

(**c**) (**d**)

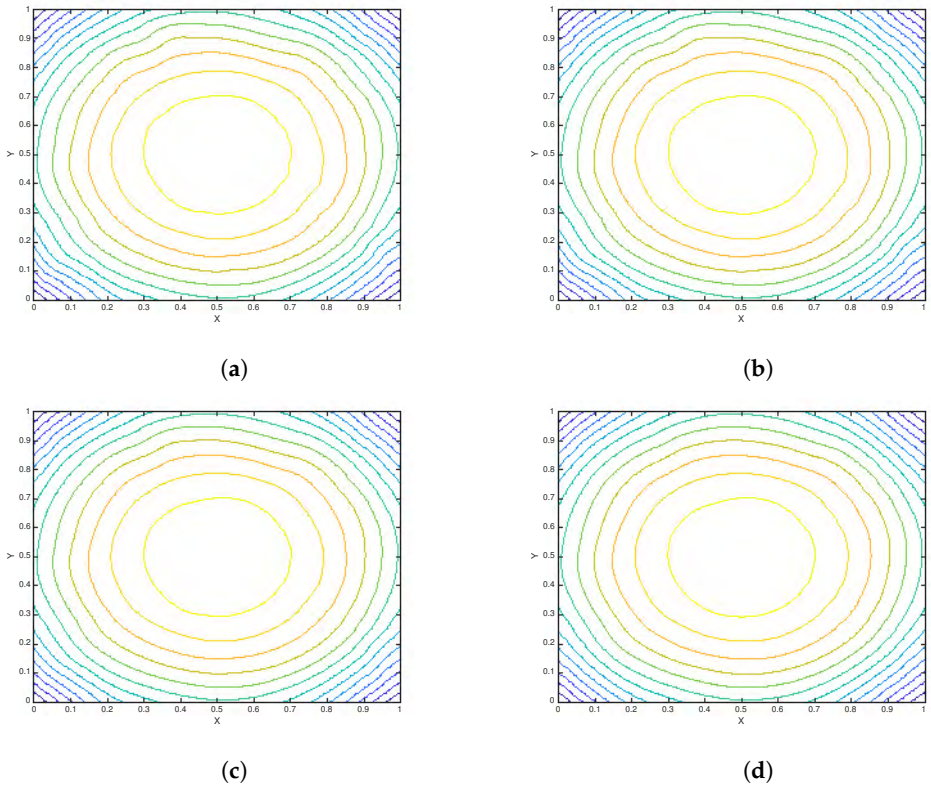
Figure 8. Contour plots based on F_1 function with 36, 65, and 100 data points when $q = 1$: (**a**) true surface; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.



(**a**) (**b**)

(**c**) (**d**)

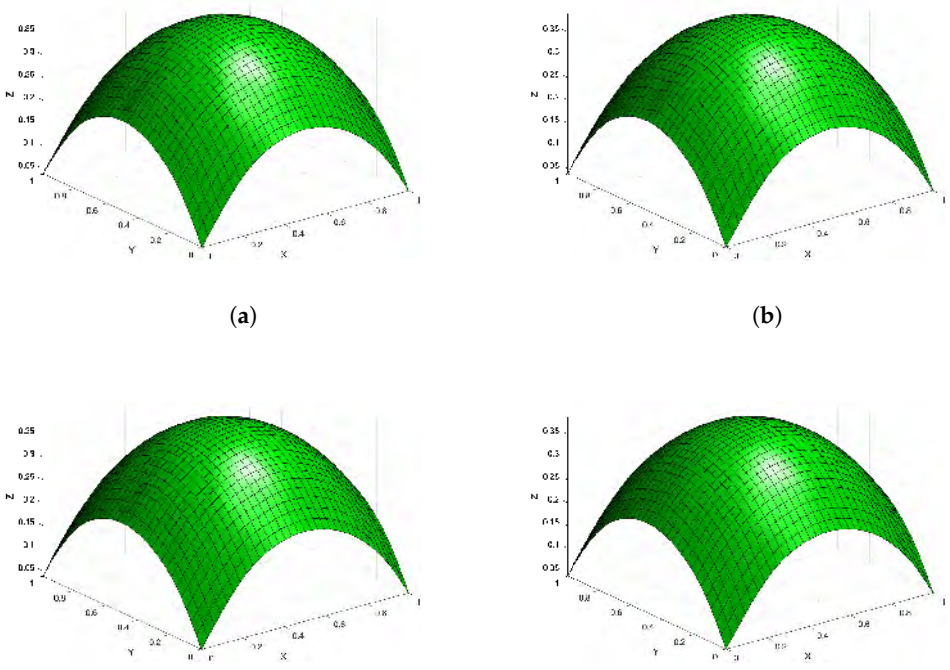
Figure 9. Surface interpolation for 36 data points based on F_1 function. (**a**) q = 0.25. (**b**) q = 0.5. (**c**) q = 0.75. (**d**) q = 1.



(**a**) (**b**)

(**c**) (**d**)

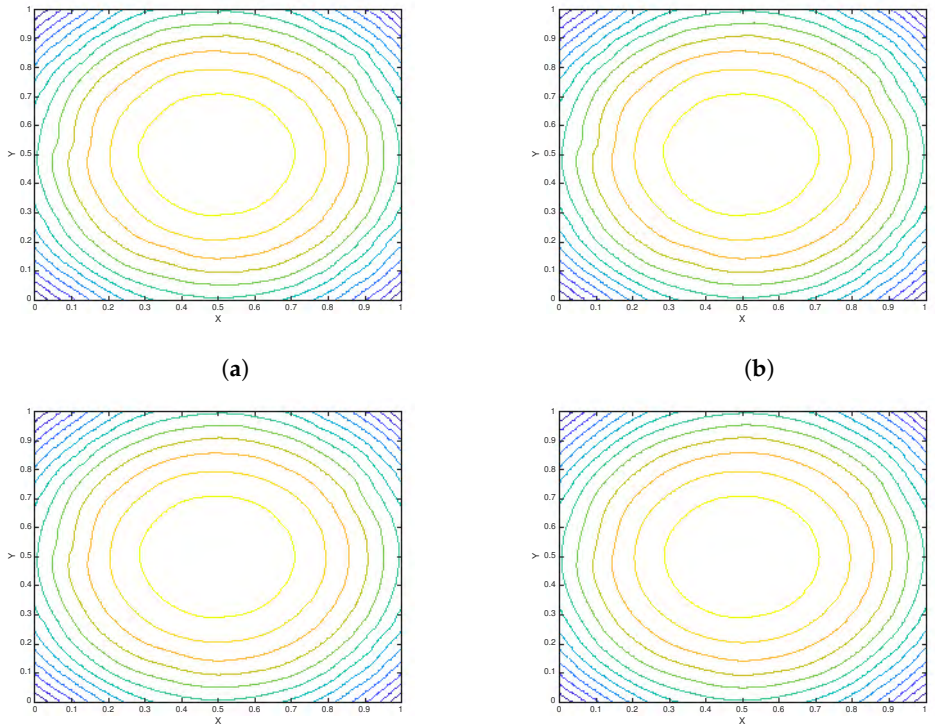
Figure 10. Contour plots for 36 data points based on F_1 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

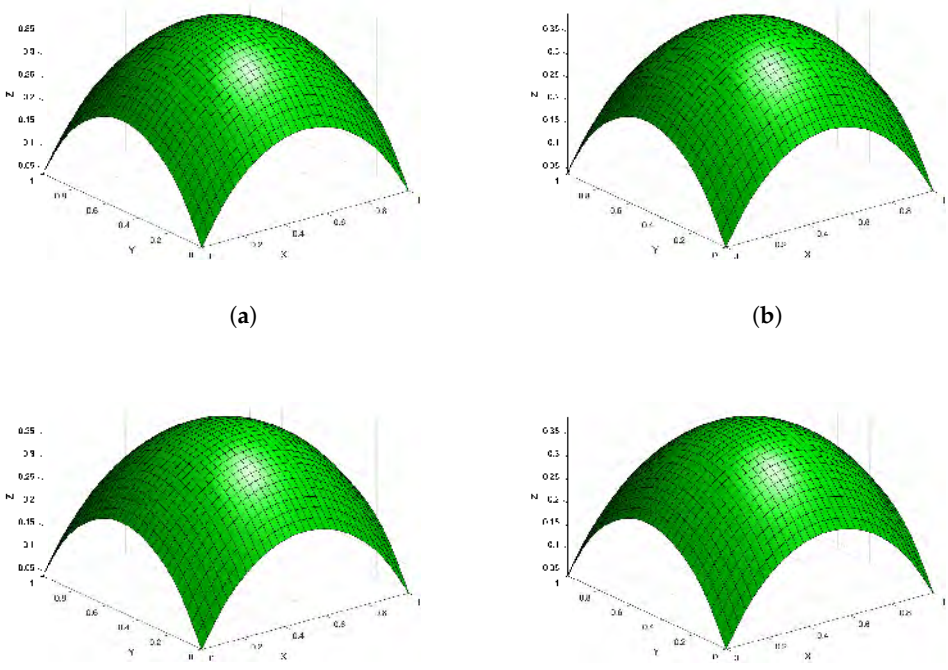
Figure 11. Surface interpolation for 65 data points based on F_1 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

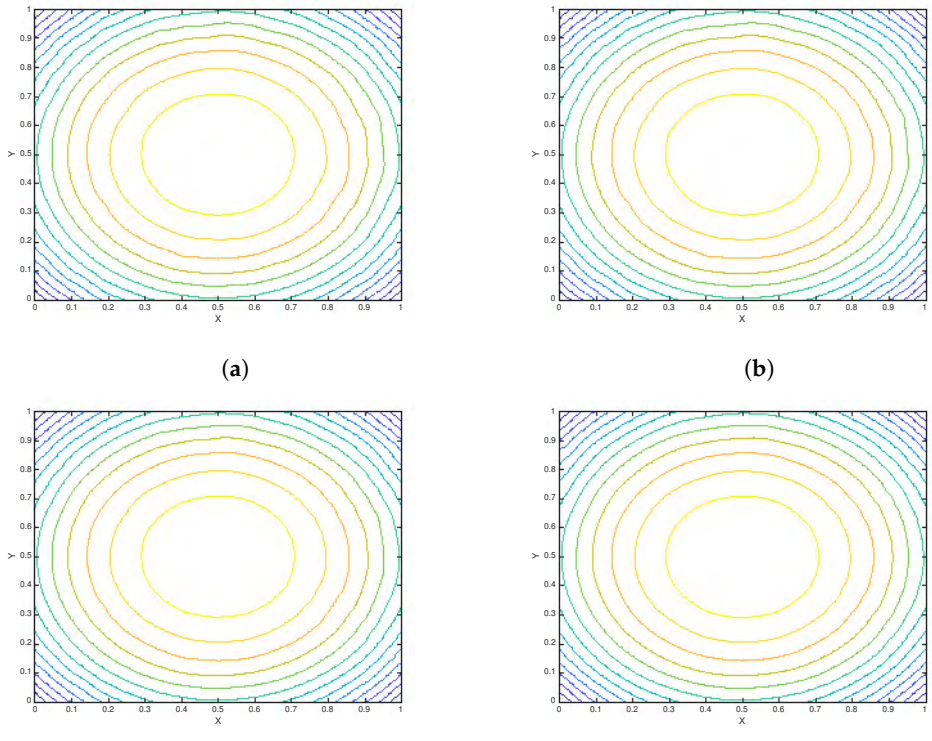
Figure 12. Contour plots for 65 data points based on F_1 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

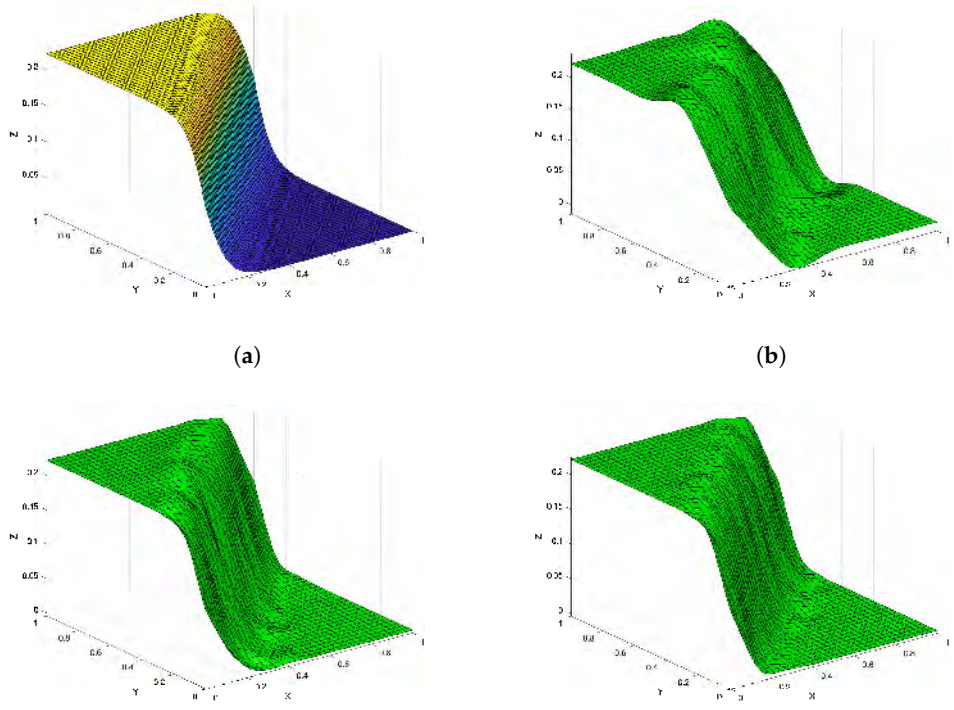
Figure 13. Surface interpolation for 100 data points based on F_1 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

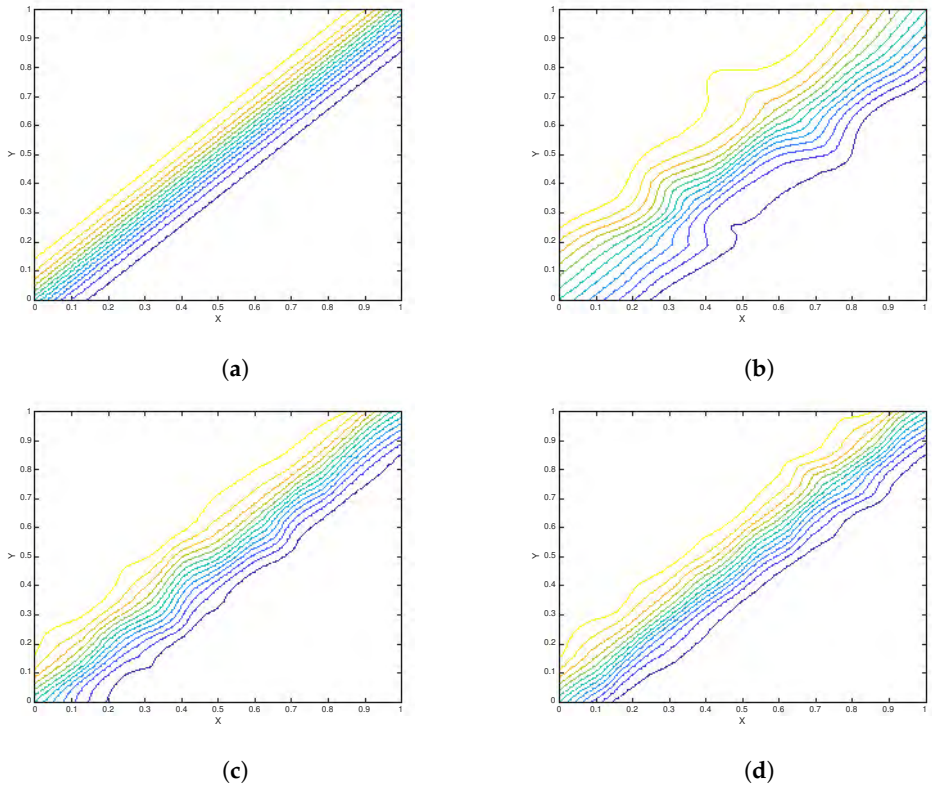
Figure 14. Contour plots for 100 data points based on F_1 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

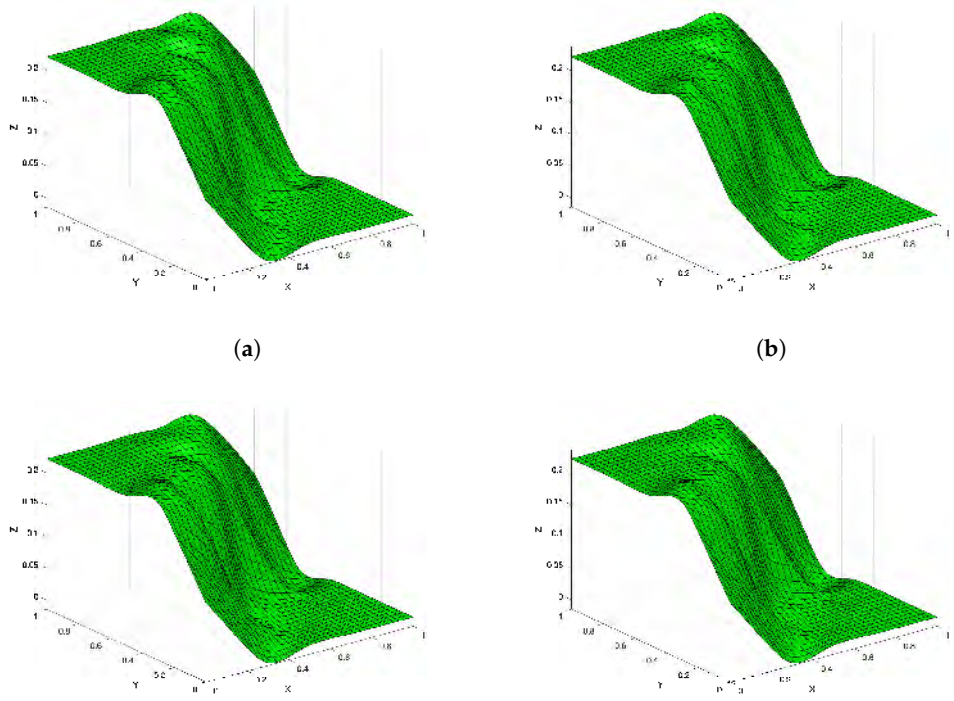
Figure 15. Surface interpolation based on F_2 function with 36, 65, and 100 data points when $q = 1$: (**a**) true surface; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.



(**a**) (**b**)

(**c**) (**d**)

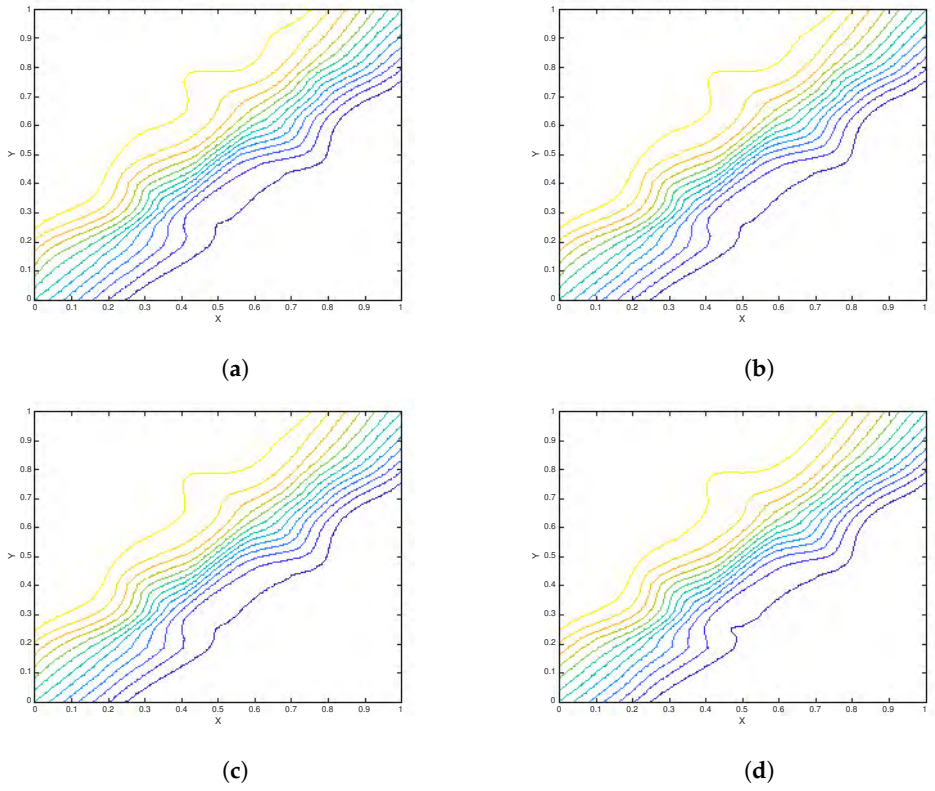
Figure 16. Contour plots based on F_2 function with 36, 65, and 100 data points when $q = 1$: (**a**) true surface; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.



(**a**) (**b**)

(**c**) (**d**)

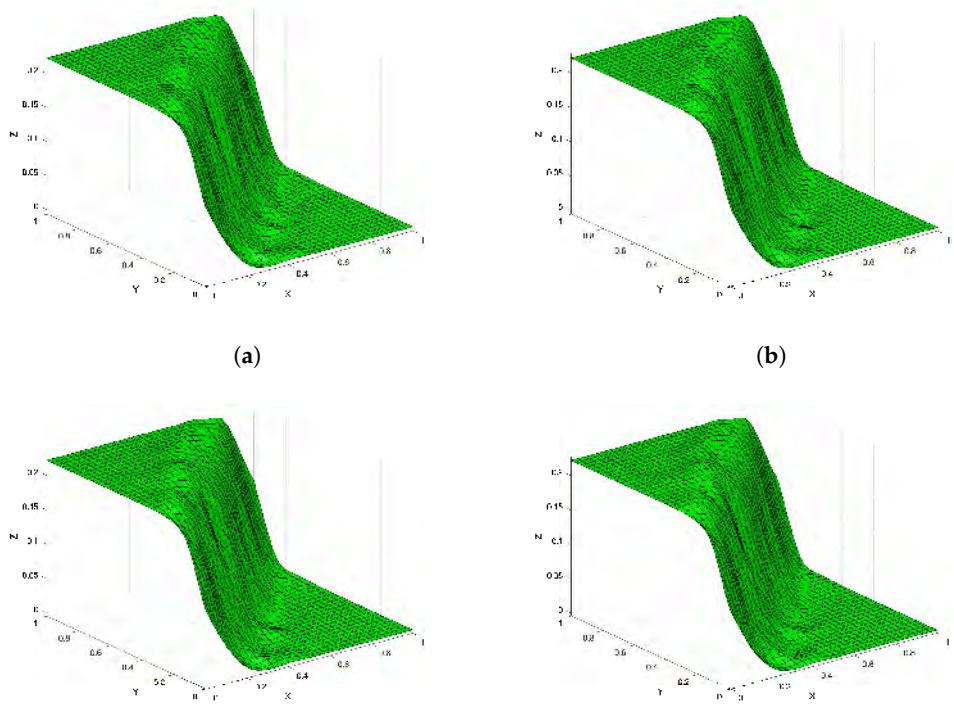
Figure 17. Surface interpolation for 36 data points based on F_2 function. (**a**) q = 0.25. (**b**) q = 0.5. (**c**) q = 0.75. (**d**) q = 1.



(**a**) (**b**)

(**c**) (**d**)

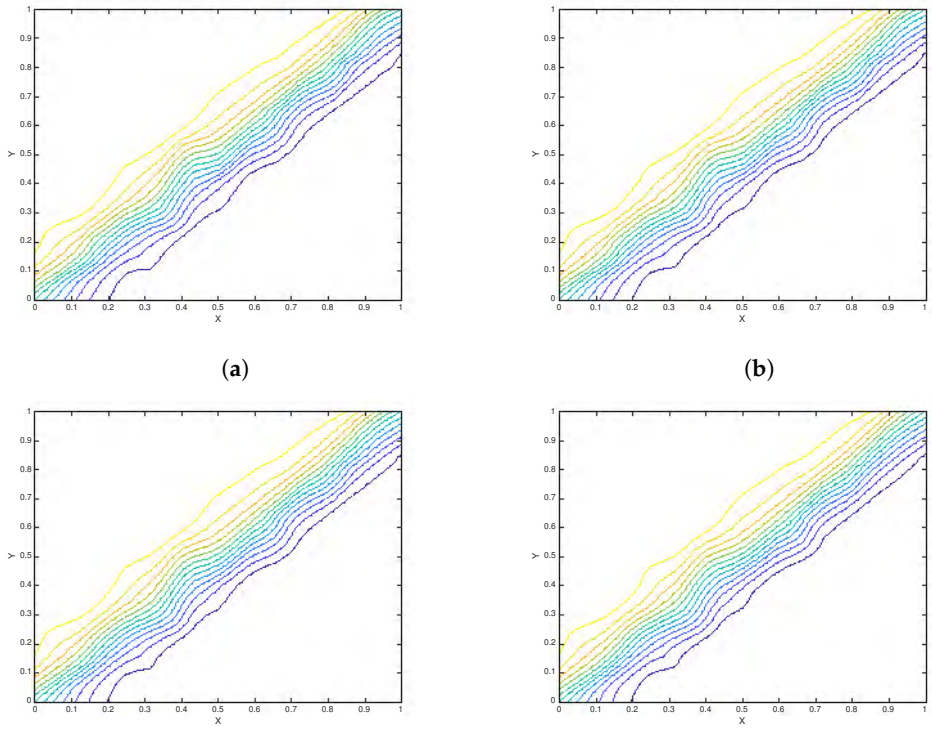
Figure 18. Contour plots for 36 data points based on F_2 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

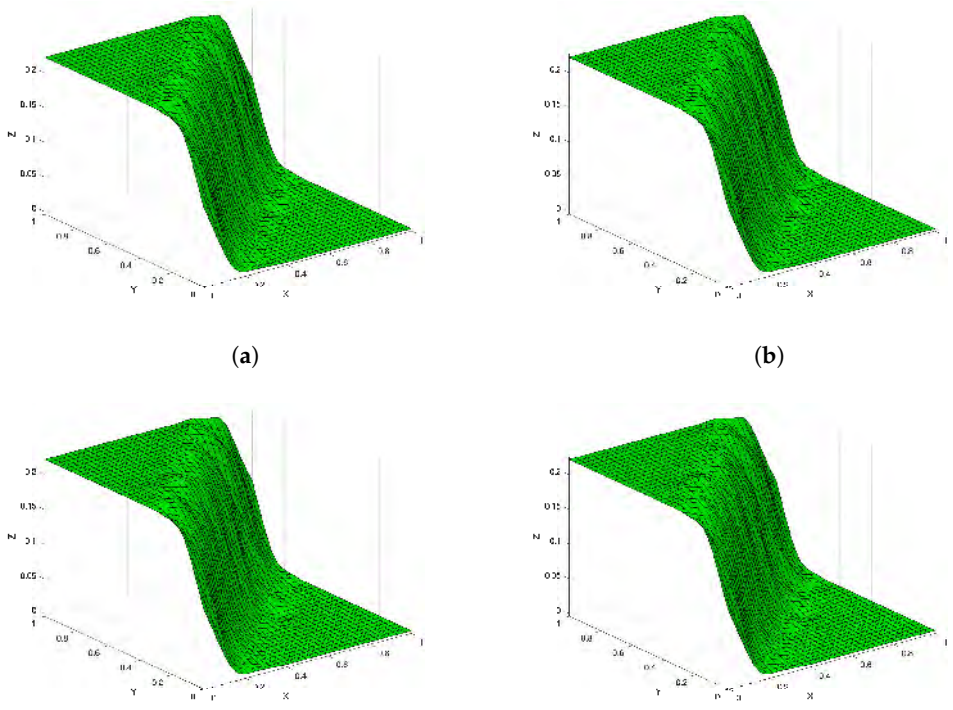
Figure 19. Surface interpolation for 65 data points based on F_2 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

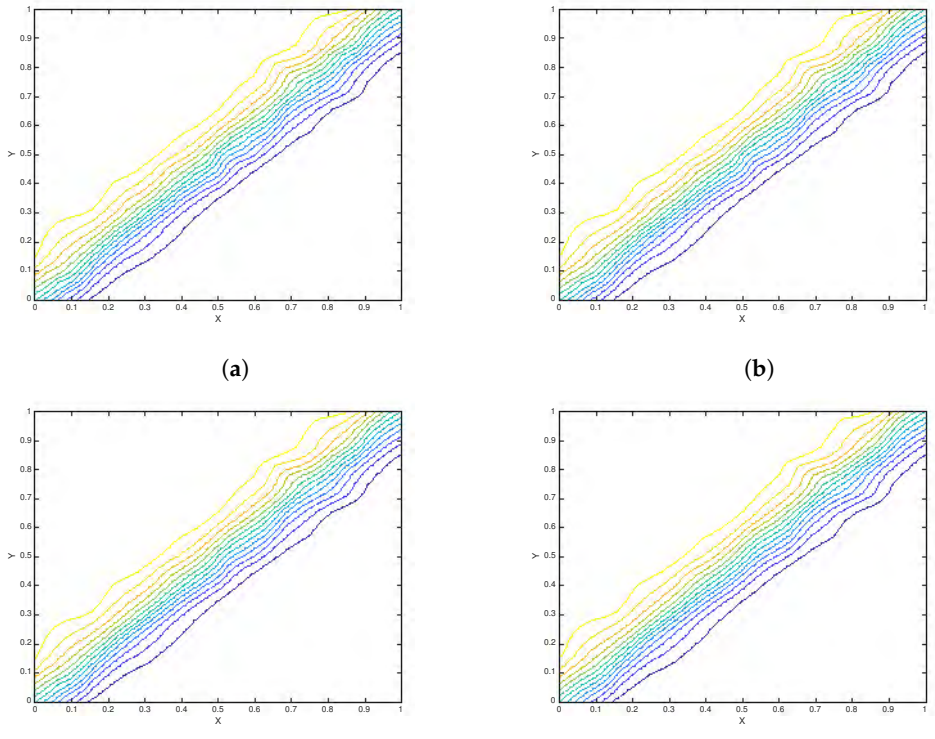
Figure 20. Contour plots for 65 data points based on F_2 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

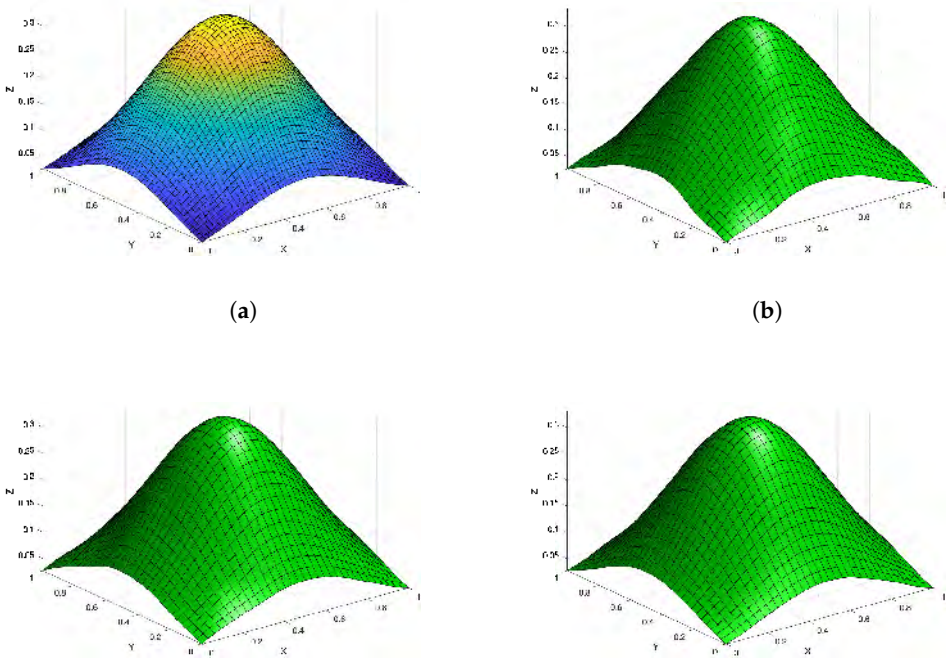
Figure 21. Surface interpolation for 100 data points based on F_2 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

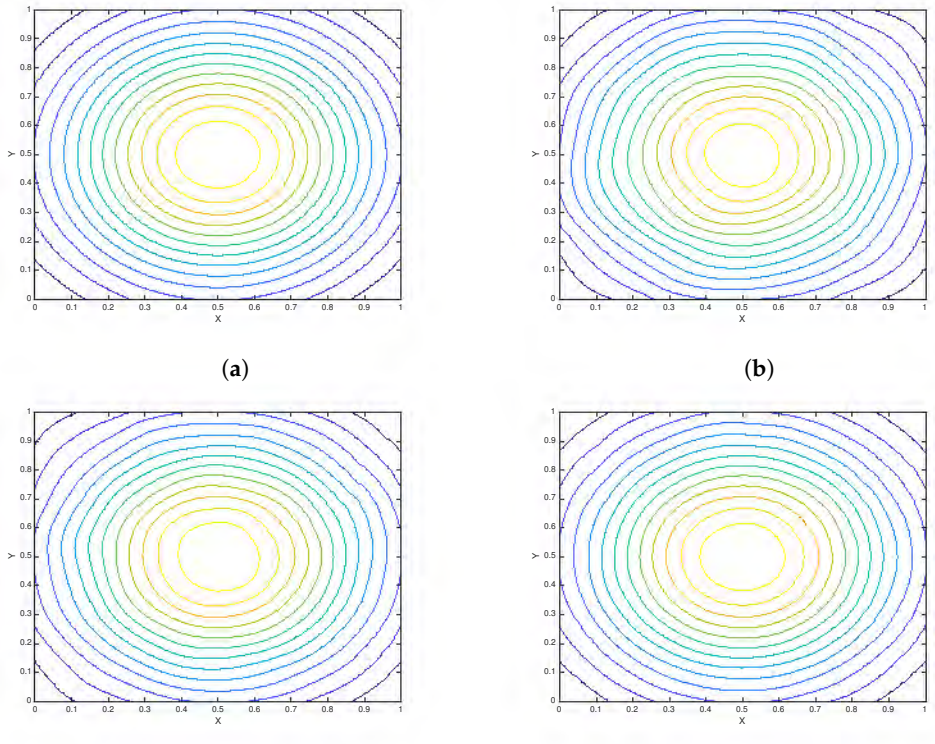
Figure 22. Contour plots for 100 data points based on F_2 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

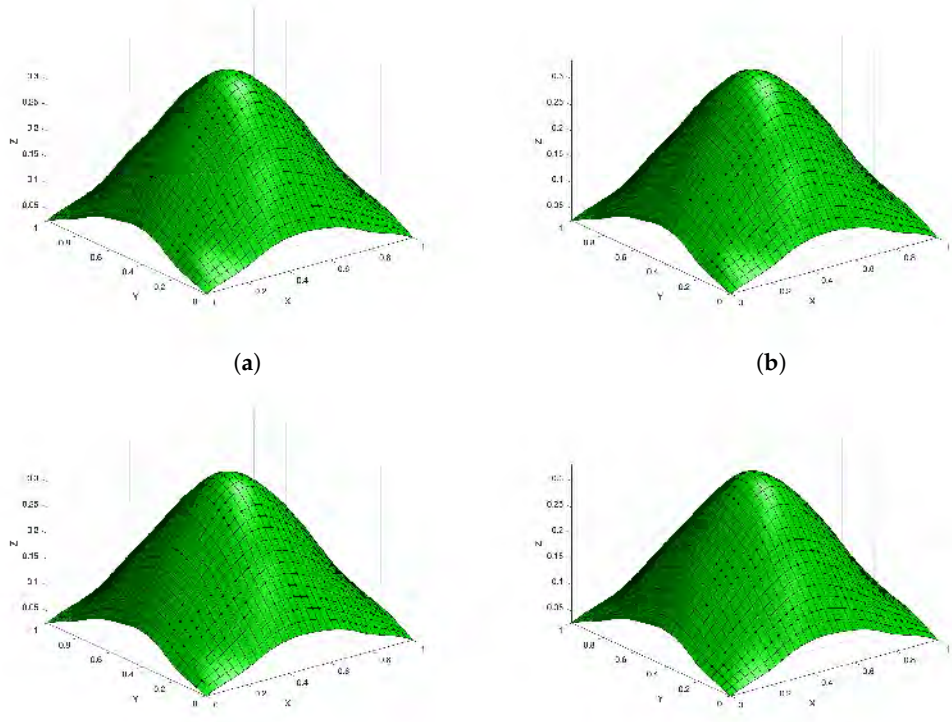
Figure 23. Surface interpolation based on F_3 function with 36, 65, and 100 data points when $q = 1$: (**a**) true surface; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.



(**a**) (**b**)

(**c**) (**d**)

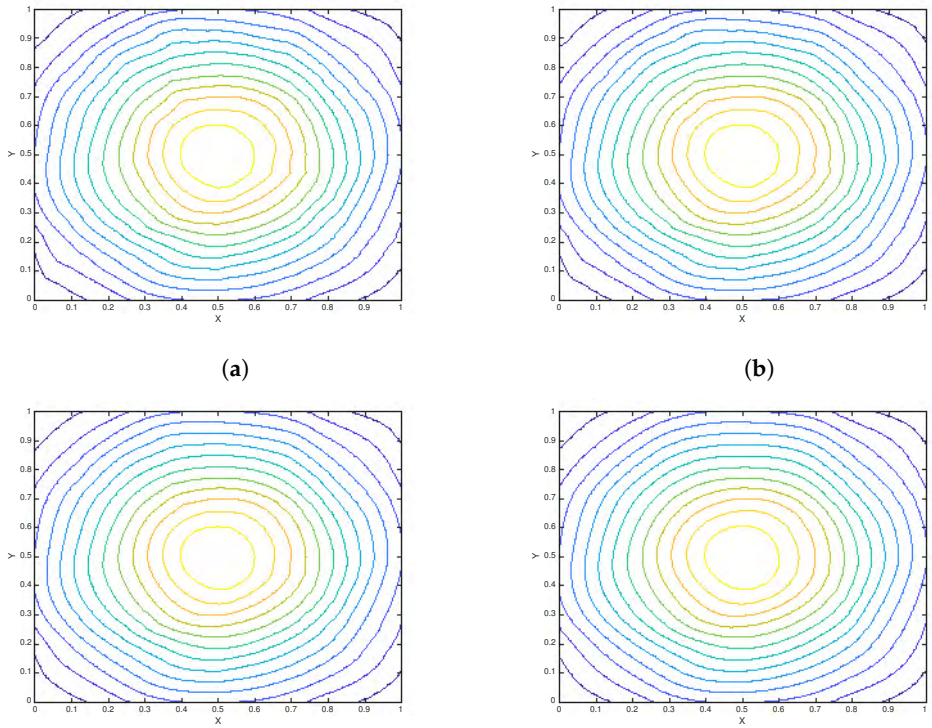
Figure 24. Contour plots based on F_3 function with 36, 65, and 100 data points when $q = 1$: (**a**) true surface; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.



(**a**) (**b**)

(**c**) (**d**)

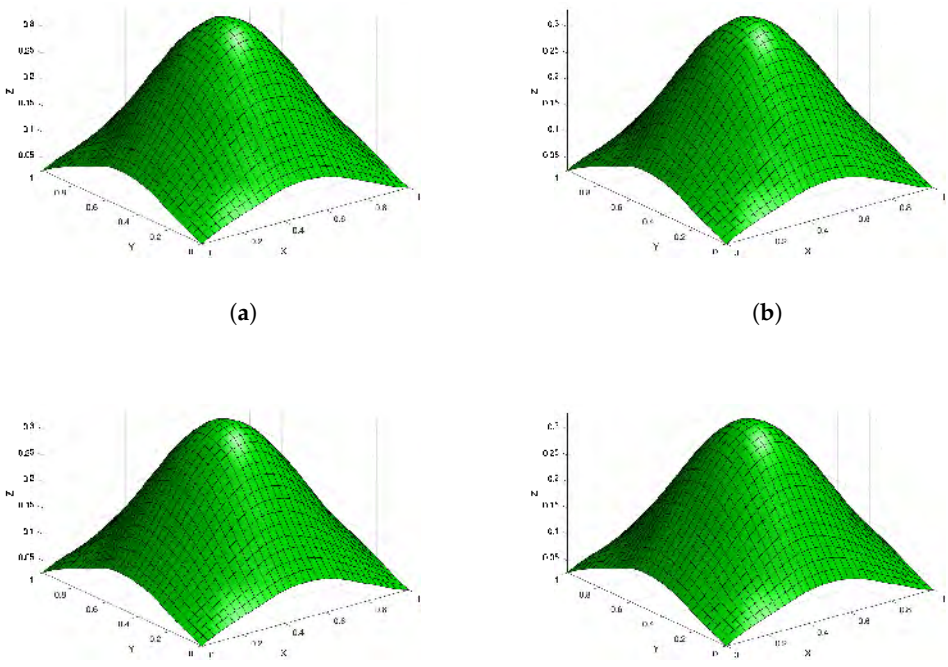
Figure 25. Surface interpolation for 36 data points based on F_3 function. (**a**) q = 0.25. (**b**) q = 0.5. (**c**) q = 0.75. (**d**) q = 1.



(**a**) (**b**)

(**c**) (**d**)

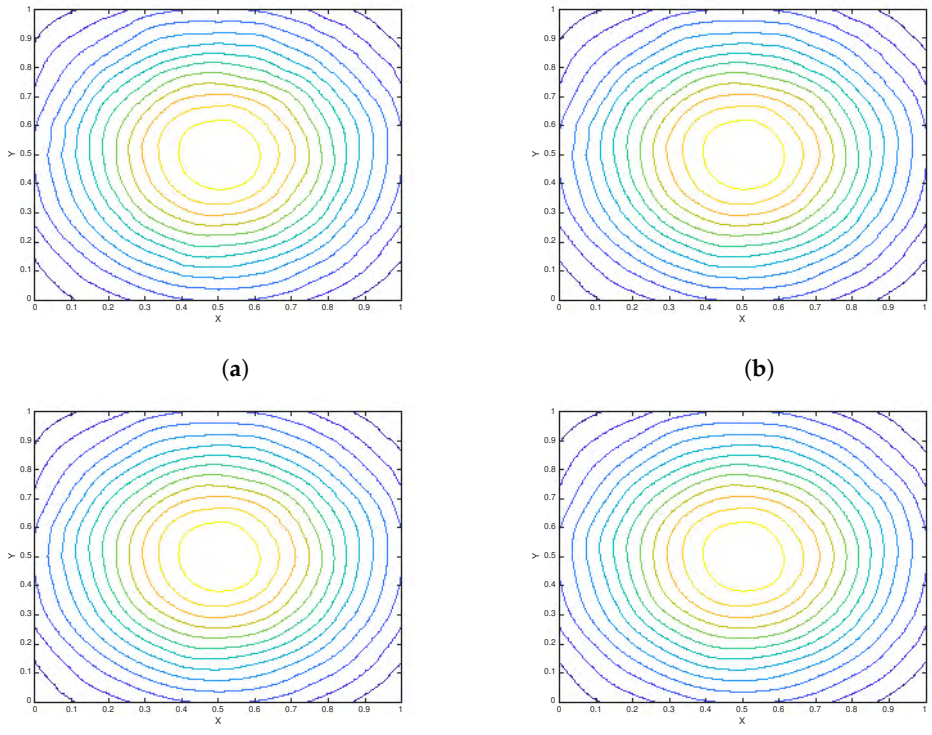
Figure 26. Contour plots for 36 data points based on F_3 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

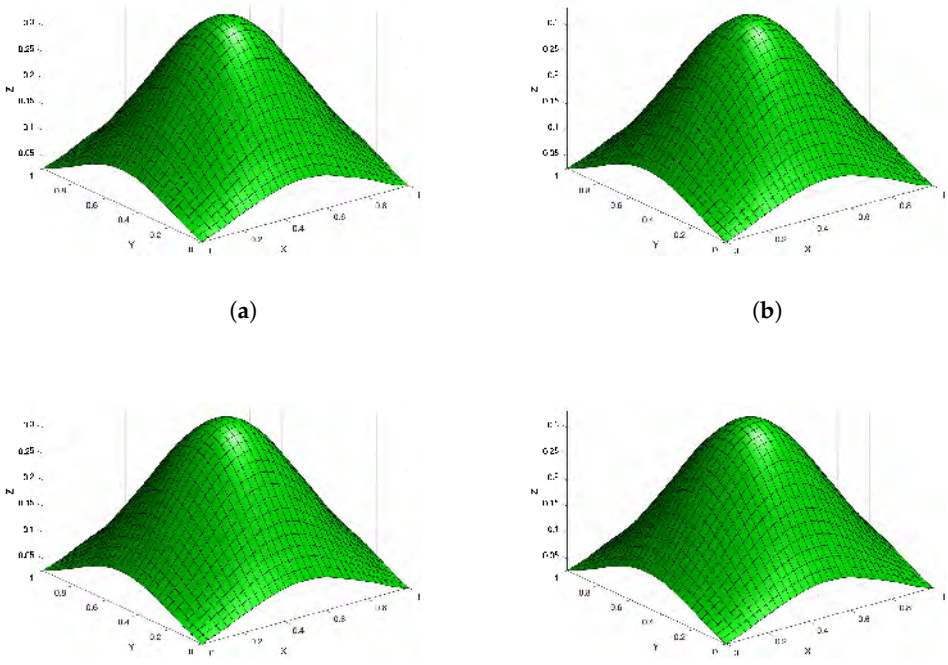
Figure 27. Surface interpolation for 65 data points based on F_3 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

Figure 28. Contour plots for 65 data points based on F_3 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

Figure 29. Surface interpolation for 100 data points based on F_3 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.

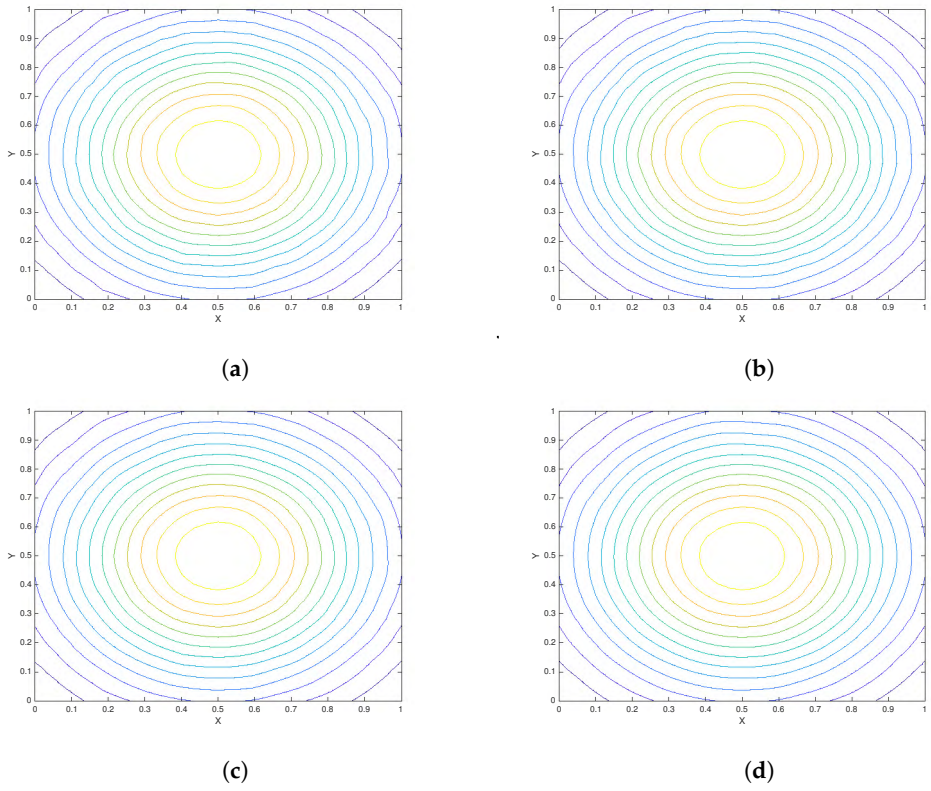


Figure 30. Contour plots for 100 data points based on F_3 function. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.

Since visualization alone is not sufficient to determine which q parameter yields the best interpolation results, error analysis was employed to provide a more robust comparison. Tables 1 and 2 present the overall measurements for 36, 65, and 100 data points with different q values for the proposed q-Bézier interpolation scheme. Based on these tables, it is evident that as the q parameter increases, there is an improvement in the interpolation quality, reflected by higher R^2 values, a reduced RMSE, and faster CPU time. Specifically, the q values of 0.75 and 1.00 show significant enhancements, indicating a trend toward the properties of the classical Bézier curve when q parameter equals 1.

Table 1. Overall error measurements for 36, 65, and 100 data points for q-Bézier with values of 0.25 and 0.50.

Function	Data Points	R^2		RMSE		CPU Time (s)	
		Q [1]	Q [2]	Q [1]	Q [2]	Q [1]	Q [2]
F_1	36	0.9974	0.9974	0.0038	0.0037	0.0981	0.1734
	65	0.9990	0.9991	0.0023	0.0023	0.2621	0.2743
	100	0.9998	0.9998	0.0011	0.0010	0.4495	0.4800
F_2	36	0.9832	0.9832	0.0129	0.0129	0.1554	0.1819
	65	0.9970	0.9971	0.0055	0.0053	0.2714	0.3210
	100	0.9986	0.9987	0.0037	0.0036	0.5213	0.4021
F_3	36	0.9973	0.9975	0.0042	0.0040	0.1336	0.1562
	65	0.9993	0.9994	0.0021	0.0020	0.2817	0.2079
	100	0.9998	0.9999	0.0011	0.0010	0.3653	0.3178

[1] $q = 0.25$; [2] $q = 0.50$.

The error analysis further supports the visual observations: the results for 100 data points consistently outperform those for 36 and 65 data points. This improvement is expected, as a higher number of data points provides a more detailed representation of the underlying function, thus enhancing the interpolation accuracy. Additionally, the CPU time analysis reveals that higher q values lead to faster computations. The combination of a higher R^2, lower RMSE, and reduced CPU time demonstrates the effectiveness of using a higher q parameter. The subsequent subsection will compare the proposed method with another state-of-the-art interpolation technique, focusing on the two best-performing q parameter values.

Table 2. Overall error measurements for 36, 65, and 100 data points for q-Bézier with values of 0.75 and 1.00.

Function	Data Points	R^2		RMSE		CPU Time (s)	
		Q [3]	Q [4]	Q [3]	Q [4]	Q [3]	Q [4]
F_1	36	0.9976	0.9977	0.0036	0.0035	0.1966	0.1893
	65	0.9991	0.9992	0.0022	0.0020	0.2424	0.2545
	100	0.9999	0.9999	0.0009	0.0009	0.3431	0.2557
F_2	36	0.9831	0.9829	0.0129	0.0130	0.2106	0.1758
	65	0.9973	0.9975	0.0051	0.0049	0.2550	0.1910
	100	0.9987	0.9988	0.0035	0.0035	0.2885	0.2807
F_3	36	0.9977	0.9978	0.0038	0.0037	0.2145	0.1850
	65	0.9995	0.9995	0.0019	0.0018	0.2301	0.2583
	100	0.9999	0.9999	0.0008	0.0007	0.3059	0.3024

[3] $q = 0.75$; [4] $q = 1.00$.

4.1. Comparison with Goodman and Said's Method

Based on the results from Tables 1 and 2, the optimal q values for the proposed interpolation method are between 0.75 and 1.00. To further validate the effectiveness of the proposed method, we compare it with the cross-derivative method of Goodman and Said [16]. Tables 3 and 4 present the overall error measurements between the proposed method and the cross derivative method for q values of 0.75 and 1.00 across three test functions.

The result indicate that the proposed method outperforms the cross-derivative method, particularly when the q value is 1.00. Specifically, the proposed method achieves higher R^2 values and lower RMSEs for all three test functions. For instance, with a q value of 1.00, the R^2 values are consistently higher, and the RMSE values are lower, indicating a more accurate fit to the data.

Although the differences in R^2 and RMSE between the proposed method and the cross-derivative method are not significant, the CPU time required for the proposed method is significantly shorter. The proposed method only takes 0.2557 s, 0.2807 s, and 0.2078 s to construct the surface interpolation for the $F_1(x, y)$, $F_2(x, y)$, and $F_3(x, y)$ functions, respectively. In contrast, the cross-derivative method requires substantially more time for similar tasks.

These findings suggest that the proposed method not only provides a high degree of accuracy but also offers significant computational advantages. Given these promising results, the next subsection will explore the application of the proposed method on real-life data.

Table 3. Overall error measurements for 36, 65, and 100 data points between the proposed method and Goodman and Said's method for q-Bézier with value of 0.75.

Function	Data Points	R^2		RMSE		CPU Time (s)	
		PS [1]	GS [2]	PS [1]	GS [2]	PS [1]	GS [2]
F_1	36	0.9976	0.9976	0.0036	0.0036	0.1966	0.1670
	65	0.9991	0.9992	0.0022	0.0021	0.2424	0.2045
	100	0.9999	0.9999	0.0009	0.0008	0.3431	0.3071

Table 3. *Cont.*

Function	Data Points	R^2		RMSE		CPU Time (s)	
		PS [1]	GS [2]	PS [1]	GS [2]	PS [1]	GS [2]
F_2	36	0.9831	0.9824	0.0129	0.0132	0.2106	0.1640
	65	0.9973	0.9973	0.0051	0.0052	0.2550	0.1768
	100	0.9987	0.9987	0.0035	0.0036	0.2885	0.2968
F_3	36	0.9977	0.9977	0.0038	0.0038	0.2145	0.1746
	65	0.9995	0.9994	0.0019	0.0019	0.2301	0.1817
	100	0.9999	0.9999	0.0008	0.0007	0.3059	0.2988

[1] Proposed method; [2] Goodman and Said's method.

Table 4. Overall error measurements for 36, 65, and 100 data points between the proposed method and Goodman and Said's method for q-Bézier with value of 1.00.

Function	Data Points	R^2		RMSE		CPU Time (s)	
		PS [1]	GS [2]	PS [1]	GS [2]	PS [1]	GS [2]
F_1	36	0.9977	0.9976	0.0035	0.0036	0.1893	0.1986
	65	0.9992	0.9992	0.0020	0.0021	0.2166	0.2374
	100	0.9999	0.9999	0.0009	0.0008	0.2557	0.3051
F_2	36	0.9829	0.9824	0.0130	0.0132	0.1758	0.1769
	65	0.9975	0.9973	0.0049	0.0052	0.1910	0.2638
	100	0.9988	0.9987	0.0035	0.0036	0.2807	0.3589
F_3	36	0.9978	0.9977	0.0037	0.0038	0.1373	0.2090
	65	0.9995	0.9994	0.0018	0.0019	0.1570	0.2496
	100	0.9999	0.9999	0.0007	0.0007	0.2078	0.4441

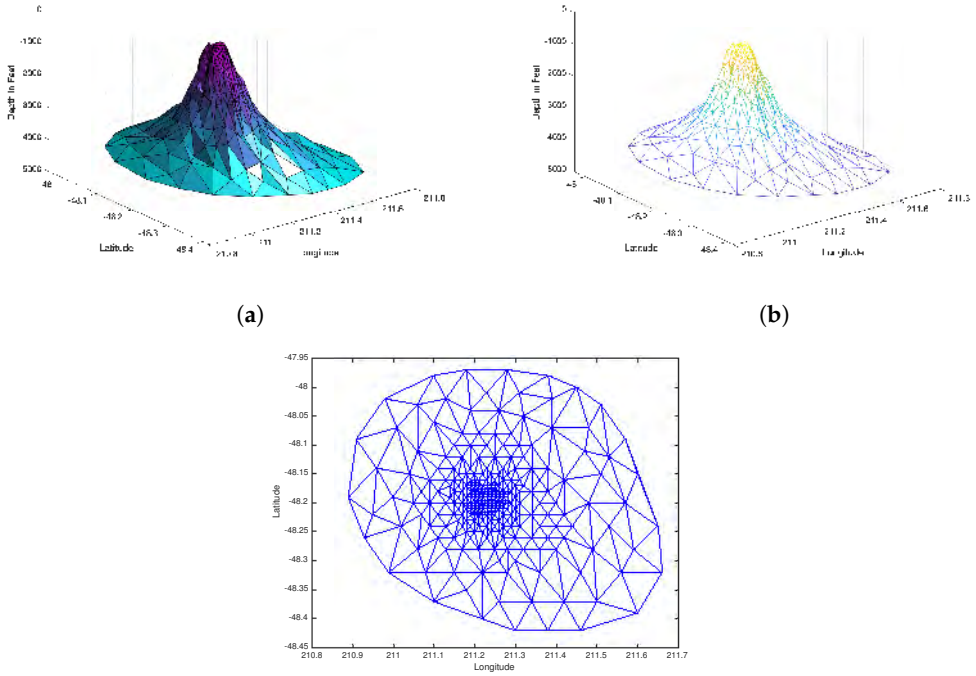
[1] Proposed method; [2] Goodman and Said's method.

4.2. Testing on Seamount Real Data Points

In this subsection, we apply the proposed method to reconstruct real-life geologic data from the Seamount dataset. This dataset, obtained from MATLAB, represents the underwater surface of a seamount located on the Louisville Ridge in the South Pacific, mapped in 1984. Figure 31 illustrates the Delaunay triangulation of the Seamount data, comprising 294 data points, along with the corresponding 3D interpolation. In Figure 32, the surface interpolation of the Seamount data points is presented for various q values. Furthermore, Figure 33 showcases the 3D interpolation derived from Figure 32.

Table 5 provides a comparative analysis of CPU times for different q values based on the average of three experimental runs. Notably, the q value of 1.00 consistently results in the shortest average time of 157.99 s. This observation aligns with previous findings, reinforcing that a q value of 1.00 not only ensures computational efficiency but also maintains interpolation accuracy.

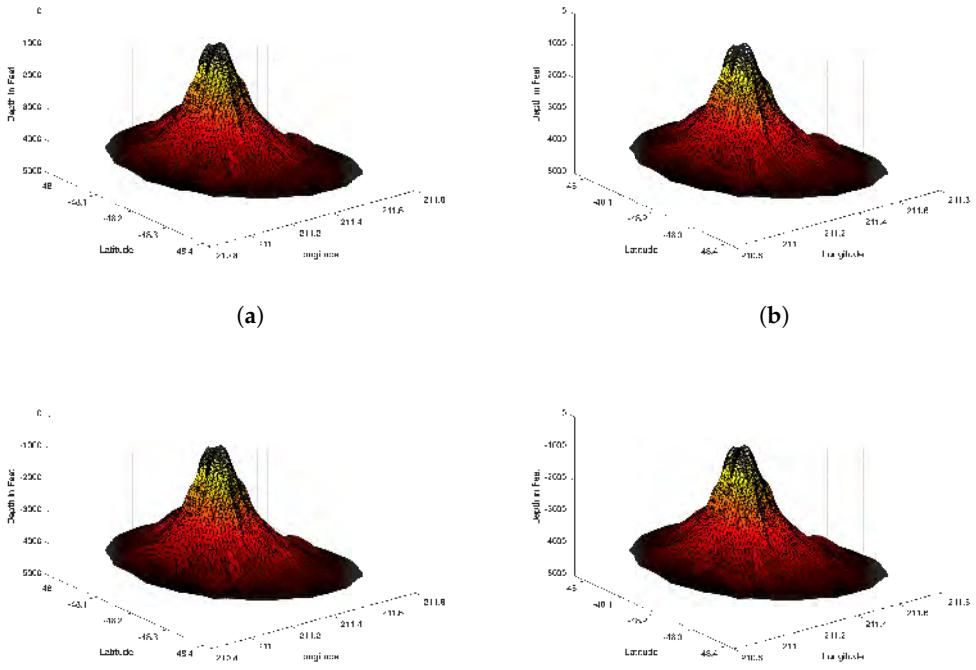
Upon examining the visualizations, it is evident that the interpolations with a q value of 1.00 yield a smoother and more accurate representation of the Seamount surface compared to other q values. The reduced computational time further emphasizes the practicality of the proposed method for large and complex datasets. The following subsections will present a more detailed comparative analysis applied to engineering-related problems.



(**a**) (**b**)

(**c**)

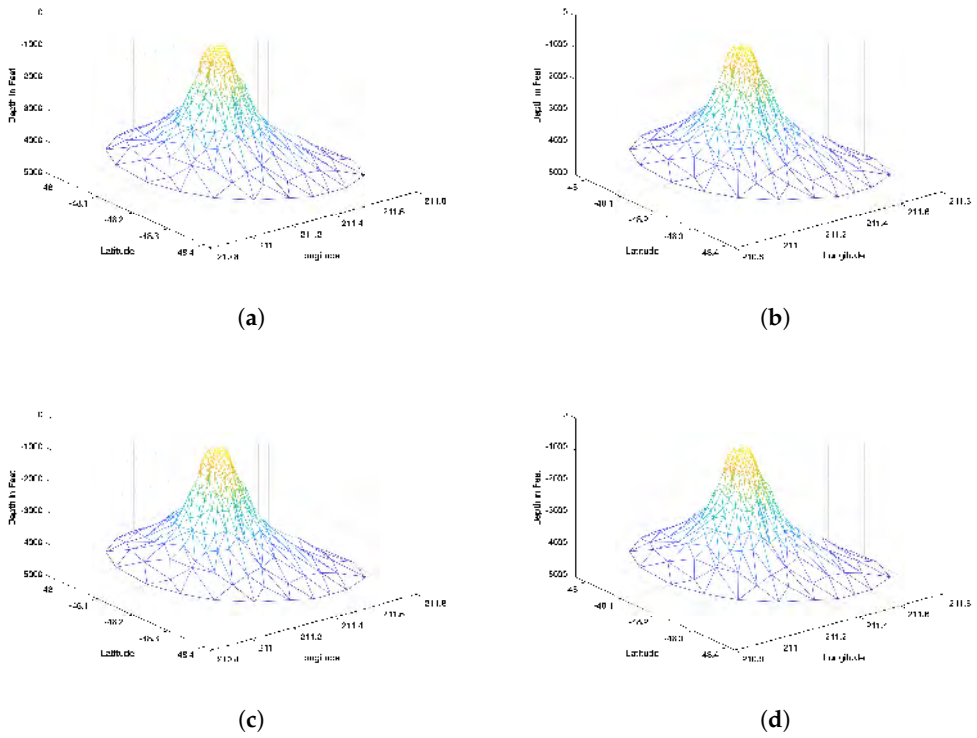
Figure 31. Example for (**a**) true surface, (**b**) 3D interpolation, and (**c**) Delaunay triangulation of Seamount data points.



(**a**) (**b**)

(**c**) (**d**)

Figure 32. Surface interpolation with different q values based on Seamount data points. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.



(**a**) (**b**)

(**c**) (**d**)

Figure 33. Three-dimensional interpolation with different q values based on Seamount data points. (**a**) $q = 0.25$. (**b**) $q = 0.5$. (**c**) $q = 0.75$. (**d**) $q = 1$.

Table 5. CPU time comparison between different q values.

q Values	CPU Time (s)			
	First Run	Second Run	Third Run	Average
0.25	163.25	159.80	158.05	160.37
0.50	160.45	158.11	157.78	158.78
0.75	157.74	157.76	159.17	158.22
1.00	157.88	158.37	157.73	157.99

4.3. Application to the Electric Potential of Two Point Charges

The scattered data interpolation method obtained in the previous subsection can be applied directly to the problem of the electric potential of two point charges proposed in [38].

The equation can be written as

$$V = \frac{1}{4\pi\kappa_0}\frac{g}{h} \tag{16}$$

where $\kappa_0 = 8.8541878 \times 10^{-12}\frac{C}{Nm^2}$ represents the permittivity constant. g is the magnitude of the charge in coulombs, whereas h is the distance from the particle in meters. Normally, the electric field of two or more particles is calculated using superposition. Based on Figure 34, the electric potential at a point due to two particles is given by

$$V = \frac{1}{4\pi\kappa_0}(\frac{g_1}{h_1} + \frac{g_2}{h_2}) \tag{17}$$

where g_1, g_2, h_1, and h_2 are the charges of the particles and the distance from the point to the corresponding particle, respectively. In taking $g_1 = 2 \times 10^{-10}C$ and $g_2 = 3 \times 10^{-10}C$, which are positioned in the x y plane at points $(0.25, 0, 0)$ and $(-0.25, 0, 0)$, the scattered data interpolation that uses the q-Bézier triangular patch can solve the proposed problem. The existing Delaunay triangulation for the proposed problem is illustrated in Figure 35 as presented by Ali et al. [39]. Figure 36 showcases the surface interpolation and its corresponding contour plot for both the true function and the existing scheme of Ali et al. [39]. However, our observation indicates that their Delaunay triangulation could be enhanced for better interpolation accuracy by creating a more scattered data distribution. Another critical observation from Figure 36 is that the method of Ali et al. does not approximate the problem accurately, as the results do not closely resemble the actual contour plot.

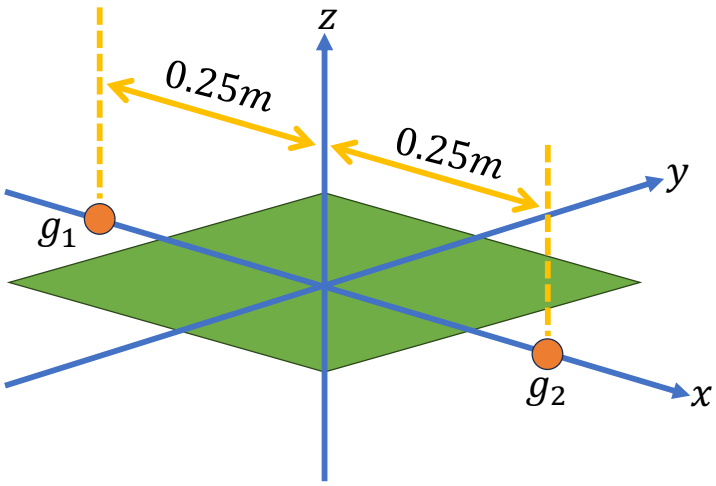


Figure 34. Example of electric potential of two point charges (Adopted from [38]).

To address these issues, we manually constructed our own Delaunay triangulation by carefully plotting scattered data with an uneven distribution, not only for 25 data points but also for 36, 65, and 100 data points, as shown in Figure 37. This extension aims to provide a more comprehensive comparative analysis of the proposed scheme. Figures 38 and 39 illustrate the surface interpolation and its contour plot when the q parameter is set to 1. Based on the figures, using 100 data points results in a smoother and more refined surface interpolation and contour plot that closely resembles the true function of the proposed problem. This pattern is consistent with previous sections of this study, where an increased number of data points provided more detailed information and features, better representing the proposed solution.

Tables 6 and 7 present the error analysis comparing Ali et al.'s method [39] and our proposed method across various q parameter values. Our proposed method achieves the best performance with 100 data points, showing an R^2 value of 0.9908 and an RMSE of 0.5753 when the q parameter is 0.25. This is particularly interesting because, in contrast to the previous sections with different functions, our proposed method performed best when the q parameter was equal to 1. This discrepancy suggests the potential for further exploration of optimal q parameter values to enhance scattered data interpolation schemes. This observation also highlights that different q values may yield the best results depending on the specific function and the nature of the real-life engineering problem being addressed.

Another noteworthy observation is that different arrangements of points, even with the same number of points, yield similar Delaunay triangulation results, as illustrated in Tables 6 and 7. The performance metrics for these different point arrangements do not show significant variation, indicating that the spatial positioning of points, given a fixed number of points, is not a crucial factor in enhancing the performance of the proposed scheme. This observation suggests that the robustness of the Delaunay triangulation method is maintained irrespective of point arrangement.

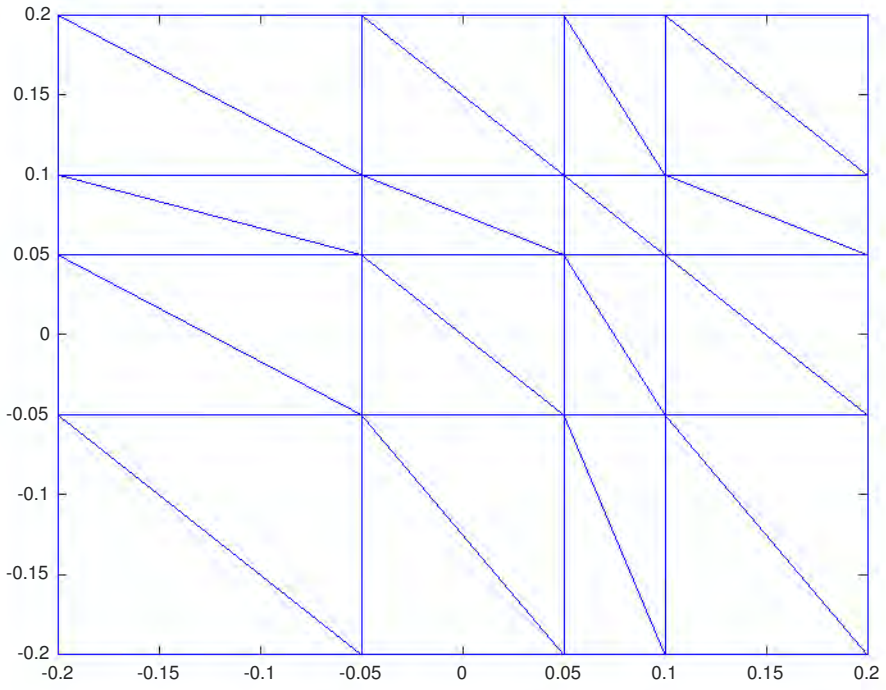


Figure 35. Delaunay triangulation of 25 data points (adopted from [39]).

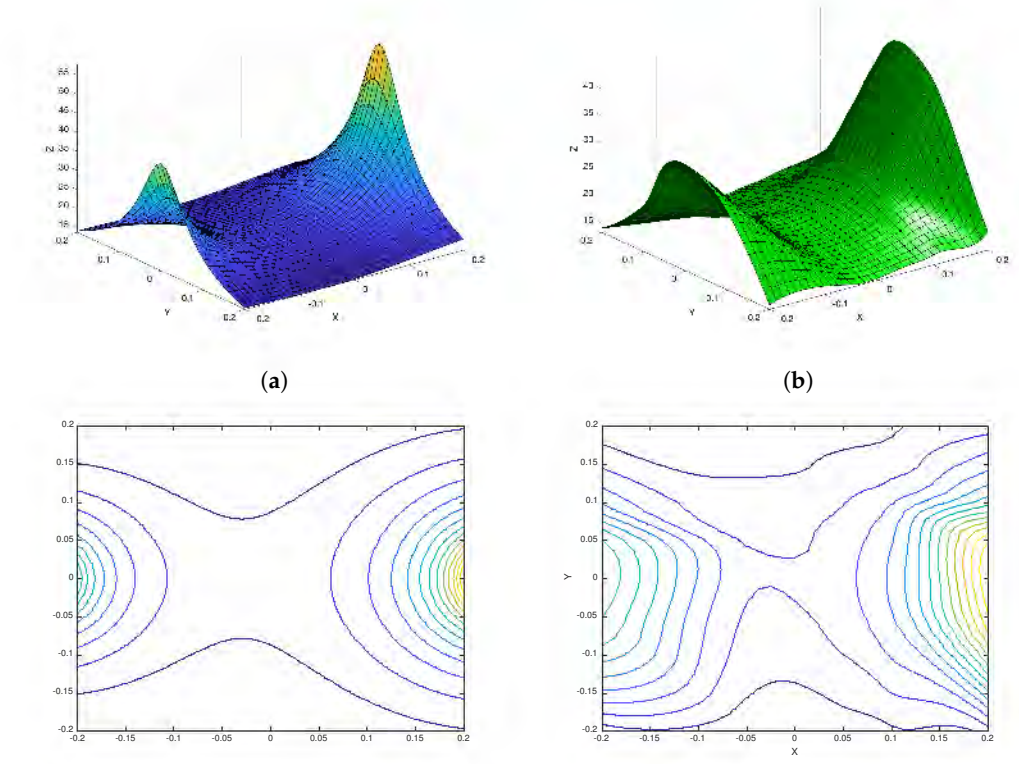
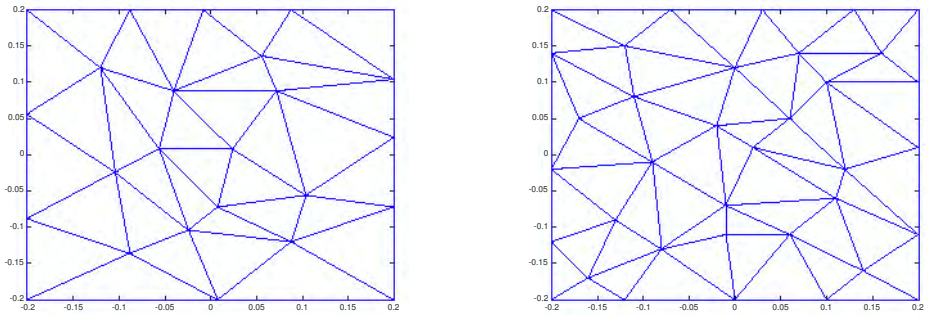
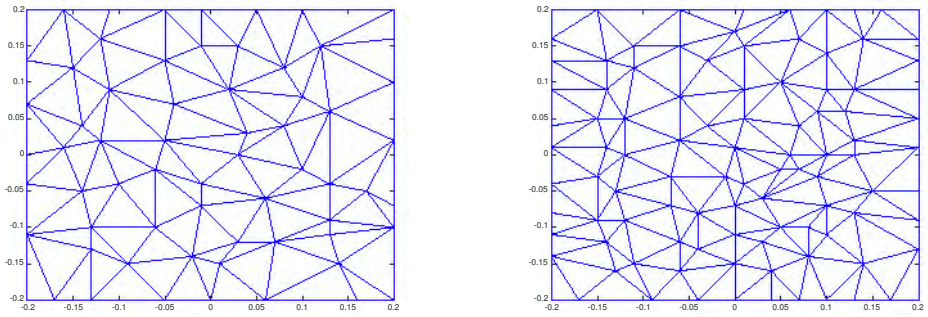


Figure 36. Surface and contour plot for (**a**) true function and (**b**) Ali et al. [39].

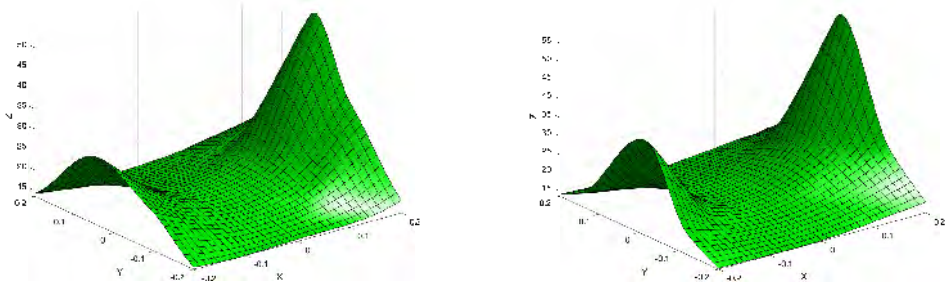


(**a**) (**b**)

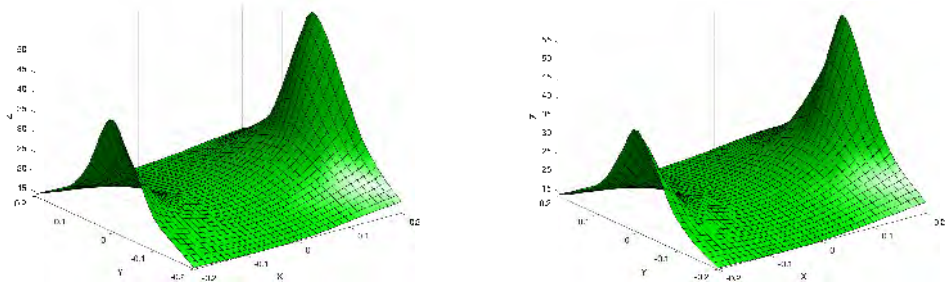


(**c**) (**d**)

Figure 37. Delaunay triangulation of 25, 35, 65, and 100 data points: (**a**) 25 data points; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.



(**a**) (**b**)



(**c**) (**d**)

Figure 38. Surface interpolation with 25, 36, 65, and 100 data points for q value of 1: (**a**) 25 data points; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.

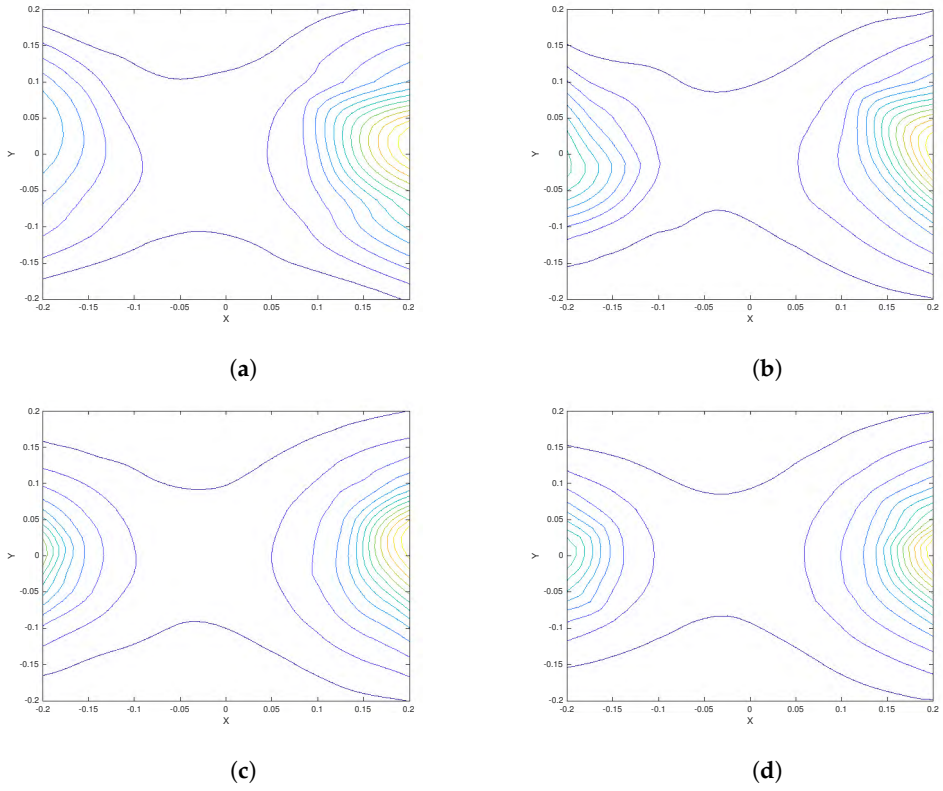


Figure 39. Contour plot with 25, 36, 65, and 100 data points for q value of 1: (**a**) 25 data points; (**b**) 36 data points; (**c**) 65 data points; (**d**) 100 data points.

Table 6. Overall error measurements for 25, 36, 65, and 100 data points for q-Bézier.

Method	Data Points	R^2		RMSE		CPU Time (s)	
		Q [1]	Q [2]	Q [1]	Q [2]	Q [1]	Q [2]
Fatin	25	0.8958	0.8934	1.9374	1.9592	0.1244	0.0989
Proposed	25	0.8868	0.8871	2.0195	2.0166	0.0743	0.1184
Proposed	36	0.9356	0.9356	1.5233	1.5225	0.1463	0.1746
Proposed	65	0.9791	0.9789	0.8674	0.8709	0.1954	0.2669
Proposed	100	0.9908	0.9907	0.5753	0.5777	0.2696	0.3106

[1] $q = 0.25$; [2] $q = 0.50$.

Table 7. Overall error measurements for 25, 36, 65, and 100 data points for q-Bézier.

Method	Data Points	R^2		RMSE		CPU Time (s)	
		Q [3]	Q [4]	Q [3]	Q [4]	Q [3]	Q [4]
Fatin	25	0.8901	0.8852	1.9898	2.0335	0.1654	0.1502
Proposed	25	0.8881	0.8893	2.0079	1.9971	0.0715	0.0995
Proposed	36	0.9361	0.9364	1.5173	1.5136	0.1160	0.1495
Proposed	65	0.9786	0.9780	0.8776	0.8898	0.1889	0.1958
Proposed	100	0.9905	0.9902	0.5836	0.5956	0.3125	0.2644

[3] $q = 0.75$; [4] $q = 1.00$.

4.4. Optimal Value of q

This section describes the optimal value of the q parameter. Previous results from visualization and error analysis indicated that the proposed scheme performs best when the q parameter is set to 1. To explore the potential for other values close to 1, we conducted additional simulations starting from a q value of 0.9 to 1.00 with a step size of 0.01. Table 8 presents the error measurements for 36, 65, and 100 data points across three test functions, ranging from $q = 0.90$ to $q = 1.00$ in increments of 0.01.

The results indicate that the R^2 and RMSE values are nearly identical for q values of 0.95, 0.96, 0.97, and 1. To facilitate visualization, we compare these four best q values using error analysis metrics in Figures 40–42. The bar chart visualization reveals no significant differences in the R^2 and RMSE values among these selected q values. Notably, the R^2 value is lowest for 36 data points based on the F_2 function, corresponding to the highest RMSE value observed for the same data points.

However, a q value of 0.96 outperforms the other three q values in terms of CPU time. Therefore, considering all error analysis metrics, a q value of 0.96 is optimal. This finding is intriguing, suggesting deviation from the classical Bézier curve typically used in state-of-the-art methods, despite $q = 0.96$ being close to 1. This presents opportunities for future research to explore optimal q values for enhancing scattered data interpolation methods.

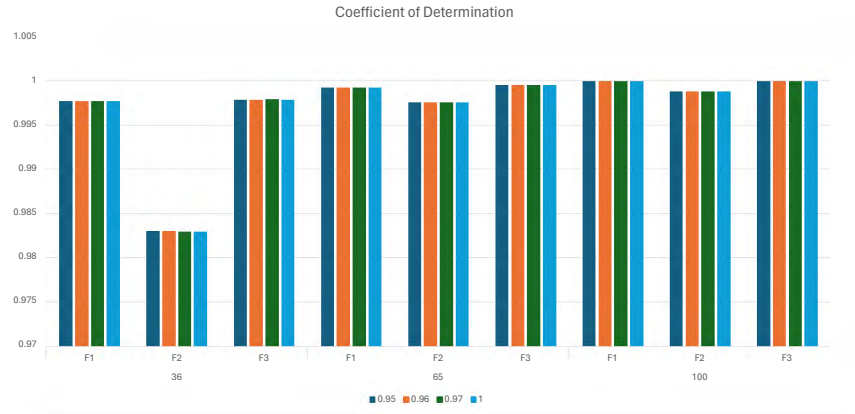


Figure 40. R^2 values for 36, 65, and 100 data points across three test functions, evaluated at four selected q values.

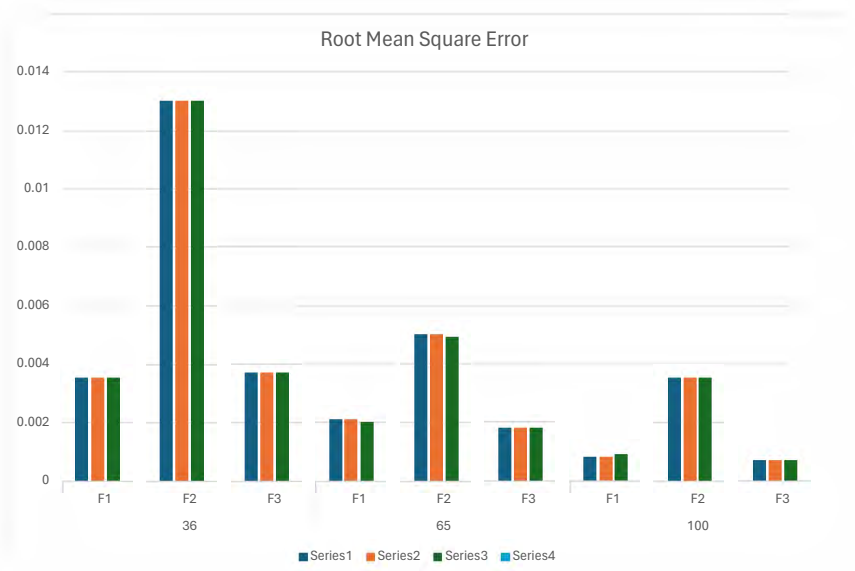


Figure 41. RMSE values for 36, 65, and 100 data points across three test functions, evaluated at four selected q values.

Table 8. Overall error measurements for 36, 65, and 100 data points for q-Bézier with values of from 0.90 to 1.00 with step size of 0.01.

q Value	Data Points								
	36			65			100		
	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3
0.90	0.9977	0.9830	0.9978	0.9992	0.9974	0.9995	0.9999	0.9988	0.9999
	0.0036	0.0130	0.0037	0.0021	0.0050	0.0018	0.0008	0.0035	0.0007
	0.1449	0.1419	0.1487	0.1752	0.2267	0.2210	0.2256	0.3326	0.2341
0.91	0.9977	0.9830	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0036	0.0130	0.0037	0.0021	0.0050	0.0018	0.0008	0.0035	0.0007
	0.1758	0.1598	0.1827	0.2049	0.2051	0.1908	0.3662	0.3654	0.3049
0.92	0.9977	0.9830	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0036	0.0130	0.0037	0.0021	0.0050	0.0018	0.0008	0.0035	0.0007
	0.1895	0.1722	0.1731	0.2196	0.2047	0.1917	0.3483	0.3253	0.3222
0.93	0.9977	0.9830	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0036	0.0130	0.0037	0.0021	0.0050	0.0018	0.0008	0.0035	0.0007
	0.1509	0.1966	0.2206	0.2154	0.2106	0.2609	0.3322	0.3873	0.3712
0.94	0.9977	0.9830	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0036	0.0130	0.0037	0.0021	0.0050	0.0018	0.0008	0.0035	0.0007
	0.1748	0.1850	0.1643	0.2268	0.2256	0.1848	0.2829	0.3162	0.3039
0.95	0.9977	0.9830	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0035	0.0130	0.0037	0.0021	0.0050	0.0018	0.0008	0.0035	0.0007
	0.1530	0.1695	0.1820	0.2018	0.2054	0.2528	0.3995	0.3927	0.3929
0.96	0.9977	0.9830	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0035	0.0130	0.0037	0.0021	0.0050	0.0018	0.0008	0.0035	0.0007
	0.1670	0.1601	0.1737	0.1840	0.2091	0.2310	0.2172	0.3576	0.2682
0.97	0.9977	0.9829	0.9979	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0035	0.0130	0.0037	0.0021	0.0050	0.0018	0.0008	0.0035	0.0007
	0.1738	0.1344	0.1882	0.1901	0.2289	0.2058	0.2811	0.5423	0.3327
0.98	0.9977	0.9829	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0035	0.0130	0.0037	0.0021	0.0050	0.0018	0.0009	0.0035	0.0007
	0.1689	0.0850	0.1773	0.2088	0.2134	0.2185	0.3095	0.2941	0.6070
0.99	0.9977	0.9829	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0035	0.0130	0.0037	0.0021	0.0050	0.0018	0.0009	0.0035	0.0007
	0.1689	0.0852	0.1309	0.1898	0.1944	0.1814	0.2792	0.2571	0.2659
1.00	0.9977	0.9829	0.9978	0.9992	0.9975	0.9995	0.9999	0.9988	0.9999
	0.0035	0.0130	0.0037	0.0020	0.0049	0.0018	0.0009	0.0035	0.0007
	0.1893	0.1758	0.1850	0.2545	0.1910	0.2583	0.2557	0.2807	0.3024

Note: For each q value, the three corresponding sub-rows represent: (i) R^2, (ii) RMSE, (iii) CPU time (in seconds).

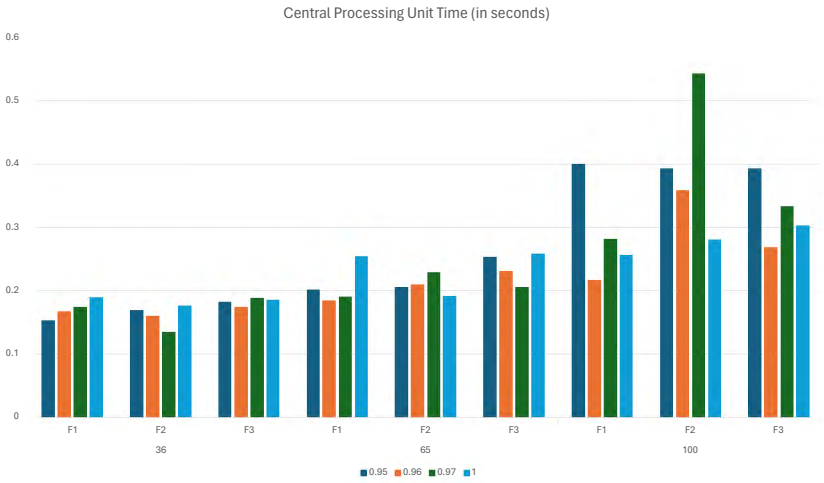


Figure 42. CPU time (in seconds) for 36, 65, and 100 data points across three test functions, evaluated at four selected q values.

5. Conclusions

In conclusion, this study successfully extends the application of q-Bézier curves to a scattered data interpolation scheme on well-known mathematical functions as well as real-life data. The proposed interpolation scheme was constructed using three local schemes that are blended through a convex combination via an efficient algorithm. The performance of the scheme was rigorously tested with 36, 65, and 100 uniformly distributed data points. Our results indicate that the proposed scheme achieves optimal visualization and error analysis outcomes when the q parameter is set between 0.75 and 1. This finding suggests that the careful tuning of the q parameter is crucial for enhancing interpolation accuracy.

The method presented in this paper is novel and still under exploration, particularly in finding the optimal q parameter. Our current approach is heuristic, and we recognize the need for a more systematic method to determine this parameter. One promising direction is to employ deep learning techniques, particularly Convolutional Neural Networks (CNNs) to optimize the q parameter. This approach involves training a CNN on a comprehensive dataset of scattered data points and their Delaunay triangulations to predict the optimal q parameter.

Additionally, our study is limited by the relatively small number of data points. Increasing the dataset size could provide more comprehensive validation of the proposed scheme and enhance its applicability to a wider range of problems.

Future research will also explore the application of the proposed scheme to image interpolation, particularly for RGB images. This extension could provide significant improvements in image processing tasks, where accurate interpolation is critical for enhancing image quality and resolution.

Author Contributions: Conceptualization, O.T.; data curation, O.T.; formal analysis, O.T. and S.A.A.K.; funding acquisition, S.A.A.K.; investigation, O.T.; methodology, O.T. and S.A.A.K.; project administration, S.A.A.K.; software, O.T. and S.A.A.K.; supervision, S.A.A.K.; validation, O.T. and S.A.A.K.; visualization, O.T. and S.A.A.K.; writing—original draft, O.T.; writing—review and editing, O.T. and S.A.A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was fully supported by the Ministry of Higher Education (MOHE) of Malaysia through Fundamental Research Grant Scheme [FRGS/1/2023/ICT06/ UMS/02/1] (New Scattered Data Interpolation Scheme Using Quasi Cubic Triangular Patches for RGB Image Interpolation) and Universiti Malaysia Sabah.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are shared in this paper.

Acknowledgments: The authors give special thanks to the Faculty of Computing and Informatics, Universiti Malaysia Sabah and School of Quantitative Sciences, College of Arts and Sciences, Universiti Utara Malaysia for the tremendous computing facility support.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RBFs	Radial Basis Functions
SNRBFs	Square Normalized Radial Basis Functions
MLS	Moving Least Square
RMSE	Root Mean Square Error
CPU	Central Processing Unit

References

1. Franke, R.; Nielson, G. Scattered data interpolation of large sets of scattered data. *Int. J. Numer. Methods Eng.* **1991**, *15*, 1–691.
2. Cavoretto, R.; De Rossi, A.; Dell'Accio, F.; Di Tommaso, F. Fast computation of triangular Shepard interpolants. *J. Comput. Appl. Math.* **2019**, *354*, 457–470. [CrossRef]
3. Liu, Z. Local multilevel scattered data interpolation. *Eng. Anal. Bound. Elem.* **2018**, *92*, 101–107. [CrossRef]

4. Le Borne, S.; Wende, M. Domain decomposition methods in scattered data interpolation with conditionally positive definite radial basis functions. *Comput. Math. Appl.* **2019**, *77*, 1178–1196. [CrossRef]
5. Skala, V.; Mourycova, E. Meshfree Interpolation of Multidimensional Time-Varying Scattered Data. *Computers* **2023**, *12*, 243. [CrossRef]
6. Joldes, G.R.; Chowdhury, H.A.; Wittek, A.; Doyle, B.; Miller, K. Modified moving least squares with polynomial bases for scattered data approximation. *Appl. Math. Comput.* **2015**, *266*, 893–902. [CrossRef]
7. Brodlie, K.W.; Asim, M.R.; Unsworth, K. Constrained visualization using the Shepard interpolation family. *Comput. Graph. Forum* **2005**, *24*, 809–820. [CrossRef]
8. Lai, M.J.; Meile, C. Scattered data interpolation with nonnegative preservation using bivariate splines and its application. *Comput. Aided Geom. Des.* **2015**, *34*, 37–49. [CrossRef]
9. Schumaker, L.L.; Speleers, H. Nonnegativity preserving macro-element interpolation of scattered data. *Comput. Aided Geom. Des.* **2010**, *27*, 245–261. [CrossRef]
10. Karim, S.A.; Saaban, A.; Hasan, M.K.; Sulaiman, J.; Hashim, I. Interpolation using cubic Bézier triangular patches. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2018**, *8*, 1746–1752. [CrossRef]
11. Karim, S.A.A.; Saaban, A.; Skala, V.; Ghaffar, A.; Nisar, K.S.; Baleanu, D. Construction of new cubic Bézier-like triangular patches with application in scattered data interpolation. *Adv. Differ. Equ.* **2020**, *2020*, 151. [CrossRef]
12. Said, H.; Wirza, R. A Cubic Ball Triangular Patch for the Scattered Data Interpolation. Master's Thesis, Pulau Pinang, Malaysia, 1994.
13. Feng, R.; Zhang, Y. Piecewise Bivariate Hermite Interpolations for large sets of scattered data. *J. Appl. Math.* **2013**, *2013*, 239703 . [CrossRef]
14. Sun, Q.; Bao, F.; Zhang, Y.; Duan, Q. A bivariate rational interpolation based on scattered data on parallel lines. *J. Vis. Commun. Image Represent.* **2013**, *24*, 75–80. [CrossRef]
15. Bozzini, M.; Lenarduzzi, L.; Rossini, M. Polyharmonic splines: An approximation method for noisy scattered data of extra-large size. *Appl. Math. Comput.* **2010**, *216*, 317–331. [CrossRef]
16. Goodman, T.; Said, H. A C^1 triangular interpolant suitable for scattered data interpolation. *Commun. Appl. Numer. Methods* **1991**, *7*, 479–485. [CrossRef]
17. Foley, T.A.; Opitz, K. Hybrid cubic Bézier triangle patches. In *Mathematical Methods in Computer Aided Geometric Design II*; Elsevier: Amsterdam, The Netherlands, 1992; pp. 275–286. [CrossRef]
18. Karim, S.A.B.A.; Saaban, A. Visualization terrain data using cubic Ball triangular patches. *MATEC Web Conf.* **2018**, *225*, 06023. [CrossRef]
19. Hussain, M.Z.; Hussain, M. C1 positivity preserving scattered data interpolation using rational Bernstein-Bézier triangular patch. *J. Appl. Math. Comput.* **2011**, *35*, 281–293. [CrossRef]
20. Chan, E.; Ong, B. Range restricted scattered data interpolation using convex combination of cubic Bézier triangles. *J. Comput. Appl. Math.* **2001**, *136*, 135–147. [CrossRef]
21. Azizan, S.; Piah, A.R.M.; Ahmad, A.M. G1 scattered data interpolation with minimized sum of squares of principal curvatures. In Proceedings of the International Conference on Computer Graphics, Imaging and Visualization (CGIV'05), Beijing, China, 26–29 July 2005.
22. Piah, A.R.M.; Saaban, A.; Majid, A.A. Range restricted positivity-preserving scattered data interpolation. *Malays. J. Fundam. Appl. Sci.* **2006**, *2*, 63–75. [CrossRef]
23. Hussain, M.; Hussain, M.Z.; Buttar, M. C^1 positive Bernstein-Bézier rational quartic interpolation. *Triangle* **2014**, *1*, 3.
24. Farin, G.E. *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed.; Morgan Kaufmann: Burlington, MA, USA, 2002; pp. 1–499. [CrossRef]
25. Franke, R. Scattered data interpolation: Tests of some methods. *Math. Comput.* **1982**, *38*, 181–200. [CrossRef]
26. Franke, R.; Nielson, G.M. Scattered data interpolation and applications: A tutorial and survey. In *Geometric Modeling: Methods and Applications*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 131–160.
27. Lodha, S.K.; Franke, R. Scattered data techniques for surfaces. In Proceedings of the Scientific Visualization Conference (Dagstuhl'97), Dagstuhl, Germany, 9–13 June 1997; IEEE: Piscataway, NJ, USA, 1997; p. 181.
28. Goldman, R.; Simeonov, P. Two essential properties of (q, h)-Bernstein–Bézier curves. *Appl. Numer. Math.* **2015**, *96*, 82–93. [CrossRef]
29. Goldman, R.; Simeonov, P.; Simsek, Y. Generating functions for the q-Bernstein bases. *SIAM J. Discret. Math.* **2014**, *28*, 1009–1025. [CrossRef]
30. Lewanowicz, S.; Woźny, P.; Area, I.; Godoy, E. Multivariate generalized Bernstein polynomials: Identities for orthogonal polynomials of two variables. *Numer. Algorithms* **2008**, *49*, 199–220. [CrossRef]
31. Oruç, H.; Phillips, G.M. q-Bernstein polynomials and Bézier curves. *J. Comput. Appl. Math.* **2003**, *151*, 1–12. [CrossRef]
32. Phillips, G.M. A de Casteljau algorithm for generalized Bernstein polynomials. *BIT Numer. Math.* **1997**, *37*, 232–236. [CrossRef]
33. Simeonov, P.; Zafiris, V.; Goldman, R. q-Blossoming: A new approach to algorithms and identities for q-Bernstein bases and q-Bézier curves. *J. Approx. Theory* **2012**, *164*, 77–104. [CrossRef]
34. Delgado, J.; Orera, H.; Peña, J.M. An evaluation algorithm for q-Bézier triangular patches formed by convex combinations. *J. Comput. Appl. Math.* **2023**, *428*, 115184. [CrossRef]

35. Schumaker, L.L. *Spline Functions: Computational Methods*; SIAM: Philadelphia, PA, USA, 2015.
36. Karim, S.A.A.; Saaban, A.; Nguyen, V.T. Scattered data interpolation using quartic triangular patch for shape-preserving interpolation and comparison with mesh-free methods. *Symmetry* **2020**, *12*, 1071. [CrossRef]
37. Fortune, S. A sweepline algorithm for Voronoi diagrams. In Proceedings of the Second Annual Symposium on Computational Geometry, Yorktown Heights, NY, USA, 2–4 June 1986; pp. 313–322. [CrossRef]
38. Gilat, A. *MATLAB: An Introduction with Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2004.
39. Ali, F.A.M.; Karim, S.A.A.; Dass, S.C.; Skala, V.; Hasan, M.K.; Hashim, I. Efficient visualization of scattered energy distribution data by using cubic timmer triangular patches. In *Energy Efficiency in Mobility Systems*; Springer: Singapore, 2020; pp. 145–180. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.