mathematics

MDPI

Article

Improved Least-Squares Progressive Iterative Approximation for Tensor Product Surfaces

Qianqian Hu *, Zhifang Wang and Ruyi Liang

School of Statistics and Mathematics, Zhejiang Gongshang University, Hangzhou 310018, China
* Correspondence: qianqian_hu@mail.zjgsu.edu.cn

Abstract: Geometric iterative methods, including progressive iterative approximation and geometric interpolation methods, are efficient for fitting a given data set. With the development of big data technology, the number of fitting data points has become massive, and the progressive iterative approximation for least-squares fitting (LSPIA) is generally applied to fit mass data. Combining the Schulz iterative method for calculating the Moore–Penrose generalized inverse matrix with the traditional LSPIA method, this paper presents an accelerated LSPIA method for tensor product surfaces and shows that the corresponding iterative surface sequence converged to the least-squares fitting surface of the given data set. The iterative format is that of a non-stationary iterative method, and the convergence rate increased rapidly as the iteration number increased. Some numerical examples are provided to illustrate that the proposed method has a faster convergence rate.

Keywords: progressive iterative approximation; least-squares fitting; tensor product surface; Schulz iteration; convergence rate

MSC: 65D17

Citation: Hu, Q.; Wang, Z.; Liang, R. Improved Least-Squares Progressive Iterative Approximation for Tensor Product Surfaces. *Mathematics* **2023**, *11*, 670. https://doi.org/10.3390/math11030670

Academic Editor: Sitnik Sergey

Received: 22 December 2022
Revised: 21 January 2023
Accepted: 23 January 2023
Published: 28 January 2023

1. Introduction

The need to apply a parametric curve or surface to fit a given set of data is an important problem encountered frequently in engineering applications, such as product styling in aerospace, computer vision, NURBS solid construction, and reverse engineering [1–4].

Progressive iterative approximation (PIA) is a simple iterative method for data fitting with intuitive geometric significance. Qi et al. presented the "profit and loss" algorithm for uniform cubic B-spline curves [5], and at the same time, de Boor proved its convergence independently [6]. Lin et al. proved that non-uniform B-spline curves and surfaces also hold this property [7]. Furthermore, Lin et al. extended this property to blending curves and surfaces with normalized totally positive (NTP) bases and named it progressive iterative approximation [8]. Over the next 15 years, this has become an attractive and promising research field, as evident from the number of publications on the PIA subject. Lin et al. provided an overview of PIA and geometric approximation methods [9]. A local progressive iterative approximation method was proposed to make data fitting more flexible by adjusting only a part of the control points [10,11]. Some improved methods have been proposed to speed up the convergence rate, such as a weighted PIA method [12], the Chebyshev accelerating PIA method [13], a preconditioned PIA method [14], and the composite Schulz–PIA method [15]. Delgado and Peña compared the convergence rates of different NTP bases and proved that the normalized B-basis had the fastest convergence rate [16]. Hamza et al. designed an implicit PIA method for implicit curve and surface reconstruction to reduce computational cost effectively [17]. In the classical PIA methods, the number of the control points needs to be equal to that of the data set, which is not suitable for fitting mass data. As we all known, the number of data set is often exceptionally large nowadays, and the LSPIA methods can handle s large set of points [18–21]. However, as the data set increases, the coefficient matrix of the least-squares fitting system may be singular.

Lin et al. proved that the LSPIA method is still convergent for singular least-squares fitting systems [22].

Tensor product surfaces are suitable for surfaces with a rectangular topology, which widely exist in engineering practice [23–25]. They can be regarded as the trajectory of the curves with variable control points in three-dimensional space. Therefore, the theoretical study on the LSPIA method for tensor product surfaces may be generalized by the curve case. An alternative explanation of the LSPIA method is to find the least-squares solution of linear systems, that is, to calculate the generalized inverse of the coefficient matrix. Its convergence rate depends on the size of the spectral radius of the iterative matrix. However, the convergence rate would greatly slow down with large-scale problems in practice. The tensor product surface cannot be treated as a curve with variable control points, although the proof of its convergence is easily generalized from the curve case. The reason is that the collocation matrix of bivariate NTP bases is a Kronecker product of two collocation matrices of NTP bases, that is, a large-scale matrix. The collocation matrix is exponentially ill-conditioned as the dimension of the collocation matrix increases [26]. To speed up the convergence rate, we combined the Schulz iterative method [27] for Moore–Penrose generalized inverse of the coefficient matrix with the traditional LSPIA method and develop an improved LSPIA format with fast convergence rate. Applying the properties of the Kronecker product and the Moore–Penrose generalized inverse, we proved that the proposed method was convergent and the limit surface was just the least-squares fitting result of the given data points.

This paper is organized as follows. Section 2 introduces the improved least-squares progressive iterative format for tensor product surfaces. The convergence of the proposed method is proved theoretically in Section 3. Then, Section 4 provides the preparatory work before the experimental examples and the improved LSPIA algorithm for fitting data points. Subsequently, some numerical experiments are presented in Section 5 to demonstrate the efficiency of the proposed algorithm by comparing the error plots and error data with those obtained with the traditional LSPIA method. Finally, the conclusions are presented in Section 6.

2. Improved Least-Squares Progressive Iterative Format for Tensor Product Surfaces

Let us consider a set of data points $\left\{\boldsymbol{Q}_{i,j}\right\}_{i=0,j=0}^{m_1,m_2}$ with the parameters $\{u_i, v_j\}_{i=0,j=0}^{m_1,m_2}$ satisfying

$$u_0 < u_1 < \cdots < u_{m_1},\ v_0 < v_1 < \cdots < v_{m_2}.$$

We arbitrarily select some data points $\{\boldsymbol{P}_{h,l}\}_{h=0,l=0}^{n_1,n_2}(n_1 < m_1, n_2 < m_2)$ from $\left\{\boldsymbol{Q}_{i,j}\right\}_{i=0,j=0}^{m_1,m_2}$ as the control points and construct an initial iterative surface $\boldsymbol{P}^0(u,v)$, i.e.,

$$\boldsymbol{P}^0(u,v) = \sum_{h=0}^{n_1}\sum_{l=0}^{n_2} B_h(u)B_l(v)\boldsymbol{P}_{h,l}^0,\ (u,v) \in [u_0, u_{m_1}] \times [v_0, v_{m_2}],$$

where $\boldsymbol{P}_{h,l}^0 = \boldsymbol{P}_{h,l}$ are the control points, and $\{B_h(u), h = 0, 1, ..., n_1\}$, $\{B_l(v), l = 0, 1, ..., n_2\}$ are the NTP bases function [8], satisfying

$$B_h(u), B_l(v) \geq 0,\ \sum_{h=0}^{n_1} B_h(u) = \sum_{l=0}^{n_2} B_l(v) = 1.$$

The collocation matrix of $\{B_h(u), h = 0, 1, ..., n_1\}$ at an increasing sequence $u_0 < u_1 < \cdots < u_{m_1}$ is

$$\boldsymbol{B}_1 = \begin{pmatrix} B_0(u_0) & B_1(u_0) & \cdots & B_{n_1}(u_0) \\ B_0(u_1) & B_1(u_1) & \cdots & B_{n_1}(u_1) \\ \vdots & \vdots & & \vdots \\ B_0(u_{m_1}) & B_1(u_{m_1}) & \cdots & B_{n_1}(u_{m_1}) \end{pmatrix}, \tag{1}$$

and the collocation matrix of $\{B_l(v), l = 0, 1, ..., n_2\}$ at $v_0 < v_1 < \cdots < v_{m_2}$ is

$$\boldsymbol{B}_2 = \begin{pmatrix} B_0(v_0) & B_1(v_0) & \cdots & B_{n_2}(v_0) \\ B_0(v_1) & B_1(v_1) & \cdots & B_{n_2}(v_1) \\ \vdots & \vdots & & \vdots \\ B_0(v_{m_2}) & B_1(v_{m_2}) & \cdots & B_{n_2}(v_{m_2}) \end{pmatrix}. \tag{2}$$

Here, $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are totally positive matrices, that is, all of their minors are nonnegative [28].

Suppose

$$\boldsymbol{Z}_1^0 = \omega_1 \boldsymbol{B}_1^T, \boldsymbol{Z}_2^0 = \omega_2 \boldsymbol{B}_2^T \tag{3}$$

where $\boldsymbol{B}_1, \boldsymbol{B}_2$ are defined as in (1) and (2), respectively, and ω_1, ω_2 satisfy the condition

$$0 < \omega_1 < 2\lambda_0^{-1},\ 0 < \omega_2 < 2\mu_0^{-1}, \tag{4}$$

where $\lambda_i (i = 0, 1, ..., n_1)$ and $\mu_j (j = 0, 1, ..., n_2)$ are the eigenvalues of $\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1$ and $\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2$ sorted in non-increasing order, respectively.

Defining

$$\delta_{i,j}^0 = \boldsymbol{Q}_{i,j} - \boldsymbol{P}^0(u_i, v_j),\ i = 0, 1, ..., m_1, j = 0, 1, ..., m_2,$$

and rewriting it into the matrix form, it yields

$$\delta^0 = \boldsymbol{Q} - \boldsymbol{B}_1 \boldsymbol{P}^0 \boldsymbol{B}_2^{\mathrm{T}}.$$

Here, $\boldsymbol{Q}$ and $\boldsymbol{P}^0$ are control points sorted in a matrix form.

For the (h, l)-th control point $\boldsymbol{P}_{h,l}^0$, we take the adjusting vector as

$$\boldsymbol{\Delta}_{hl}^0 = \sum_{i=0}^{m_1} \sum_{j=0}^{m_2} (\boldsymbol{Z}_1^1)_{h,i} \boldsymbol{\delta}_{i,j}^0 (\boldsymbol{Z}_2^1)_{l,j},\ h = 0, 1, ..., n_1, l = 0, 1, ..., n_2, \tag{5}$$

where $(\boldsymbol{Z}_1^1)_{h,i}$ is the (h, i)-th element of $\boldsymbol{Z}_1^1$, and $(\boldsymbol{Z}_2^1)_{l,j}$ is the (l, j)-th element of $\boldsymbol{Z}_2^1$. Here, $\boldsymbol{Z}_1^1$ and $\boldsymbol{Z}_2^1$ are defined by

$$\boldsymbol{Z}_1^1 = (2\boldsymbol{I}_1 - \boldsymbol{Z}_1^0 \boldsymbol{B}_1)\boldsymbol{Z}_1^0,\ \boldsymbol{Z}_2^1 = (2\boldsymbol{I}_2 - \boldsymbol{Z}_2^0 \boldsymbol{B}_2)\boldsymbol{Z}_2^0 \tag{6}$$

where $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are the identity matrices of order $(n_1 + 1)$ and $(n_2 + 1)$, respectively, $\boldsymbol{Z}_1^0$ and $\boldsymbol{Z}_2^0$ are defined as in (3).

Rewriting the adjusting vectors (5) into the matrix form, it yields

$$\boldsymbol{\Delta}^0 = \boldsymbol{Z}_1^1 \cdot \boldsymbol{\delta}^0 (\boldsymbol{Z}_2^1)^{\mathrm{T}}.$$

For each control point $\boldsymbol{P}_{h,l}^0$, moving it along the adjusting vector $\Delta_{h,l}^0$, we have

$$\boldsymbol{P}^1 = \boldsymbol{P}^0 + \boldsymbol{\Delta}^0.$$

Then, we can obtain the first iterative surface as

$$\boldsymbol{P}^1(u,v) = \sum_{h=0}^{n_1}\sum_{l=0}^{n_2} B_h(u)B_l(v)\boldsymbol{P}_{h,l}^1.$$

Similarly, we can obtain the iterative surface $\boldsymbol{P}^k(u,v)$ after the k-th iteration. According to the iterative format

$$\begin{cases} \boldsymbol{\delta}^k = \boldsymbol{Q} - \boldsymbol{B}_1\boldsymbol{P}^k\boldsymbol{B}_2^T \\ \boldsymbol{Z}_1^{k+1} = (2\boldsymbol{I}_1 - \boldsymbol{Z}_1^k\boldsymbol{B}_1)\boldsymbol{Z}_1^k \\ \boldsymbol{Z}_2^{k+1} = (2\boldsymbol{I}_2 - \boldsymbol{Z}_2^k\boldsymbol{B}_2)\boldsymbol{Z}_2^k \\ \boldsymbol{\Delta}^k = \boldsymbol{Z}_1^{k+1}\boldsymbol{\delta}^k(\boldsymbol{Z}_2^{k+1})^{\mathrm{T}} \\ \boldsymbol{P}^{k+1} = \boldsymbol{P}^k + \boldsymbol{\Delta}^k \end{cases}, \tag{7}$$

the $(k+1)$-*st* iterative tensor product surface

$$\boldsymbol{P}^{k+1}(u,v) = \sum_{h=0}^{n_1}\sum_{l=0}^{n_2} B_h(u)B_l(v)\boldsymbol{P}_{h,l}^{k+1}, \tag{8}$$

is generated.

The iterative process is repeated, and we obtain an iterative surface sequence $\left\{\boldsymbol{P}^k(u,v), k = 0,1,...\right\}$.

3. Convergence Analysis for the Improved LSPIA

In this section, if the collocation matrices $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are of full column rank, we present Theorem 1 to prove that the surface sequence $\left\{\boldsymbol{P}^k(u,v), k = 0,1,...\right\}$ generated by the iterative format (7) is convergent, and the limit surface is just the least-squares fitting of the given data points $\left\{\boldsymbol{Q}_{i,j}\right\}_{i=0,j=0}^{m_1,m_2}$. The initial tensor product surface has the least-squares progressive-iterative approximation property.

An alternative explanation of LSPIA is to find the least-squares solution of the matrix equation

$$\boldsymbol{B}_1\boldsymbol{X}\boldsymbol{B}_2^{\mathrm{T}} = \boldsymbol{Q}. \tag{9}$$

If the collocation matrices $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are of full column rank, the above equation is equivalent to

$$\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1\boldsymbol{X}\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2 = \boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{B}_2.$$

Minimizing the L_2-norm of the residual matrix, the solution for the matrix Equation (8) is the control points of the least-squares fitting surface to the given data points $\boldsymbol{Q}$.

A sufficient condition for the matrix Equation (9) with one solution is [29]

$$\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{+}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{B}_2(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{+}(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2) = \boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{B}_2,$$

where $(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{+}$ is the Moore–Penrose generalized inverse of $\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1$, and $(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{+}$ is equal to $(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}$, when $\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1$ is non-singular, and $(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{+}$ is treated analogously. The least-squares solution for the matrix Equation (9) is

$$\boldsymbol{X} = (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{B}_2(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1} \tag{10}$$

Lemma 1. *When the collocation matrices $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are of full column rank, $\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1$ and $\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2$ are positive definite matrices. Furthermore, they are non-singular and have positive eigenvalues.*

Proof. Consider that $\text{rank}(\boldsymbol{B}_1) = n_1 + 1$. Therefore, the homogeneous linear equation $\boldsymbol{B}_1\boldsymbol{x} = \boldsymbol{0}$ has only zero solution. For any non-zero column vector x in $(n_1 + 1)$ -dimension, there is $\boldsymbol{B}_1\boldsymbol{x} \neq \boldsymbol{0}$, and

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1\boldsymbol{x} = \|\boldsymbol{B}_1\boldsymbol{x}\|_2^2 > 0.$$

Therefore, $\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1$ is a positive definite matrix. It is non-singular and has positive eigenvalues. $\boldsymbol{B}_2$ is handled in a similar fashion. Thus, the Lemma is proved. □

Lemma 2. *For all the matrices Z_1^k, Z_2^k defined as in (7), if $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are of full column rank, then we have*

$$\boldsymbol{Z}_1^k\boldsymbol{Q}(\boldsymbol{Z}_2^k)^{\mathrm{T}} = (\boldsymbol{Z}_1^k\boldsymbol{B}_1)(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{Z}_2^k\boldsymbol{B}_2)(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}} \quad k = 0, 1, 2, ..., \tag{11}$$

where $\boldsymbol{Z}_1^0, \boldsymbol{Z}_2^0$ are defined by (3).

Proof. Mathematical induction is used. According to Lemma 1, when $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are of full column rank, $\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1$ and $\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2$ are both non-singular. Then, applying (3) and (10), it yields

$$\begin{aligned}&\boldsymbol{Z}_1^0\boldsymbol{B}_1(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{Z}_2^k\boldsymbol{B}_2)(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}\\ &= \omega_1\omega_2\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{B}_2(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2) = \omega_1\omega_2\boldsymbol{B}_1^{T}\boldsymbol{Q}\boldsymbol{B}_2 = \boldsymbol{Z}_1^0\boldsymbol{Q}(\boldsymbol{Z}_2^0)^{\mathrm{T}}.\end{aligned}$$

Then (11) is true when $k = 0$. Assume that Lemma 2 is true for the case of k, i.e.,

$$\boldsymbol{Z}_1^k\boldsymbol{Q}(\boldsymbol{Z}_2^k)^{\mathrm{T}} = (\boldsymbol{Z}_1^k\boldsymbol{B}_1)(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{Z}_2^k\boldsymbol{B}_2)(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}.$$

Applying to the iterative formulae for $\boldsymbol{Z}_1^{k+1}$ and $\boldsymbol{Z}_2^{k+1}$ in (6), we have

$$\begin{aligned}&(\boldsymbol{Z}_1^{k+1}\boldsymbol{B}_1)(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{Z}_2^{k+1}\boldsymbol{B}_2)(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}.\\ &\quad= (2\boldsymbol{I}_1 - \boldsymbol{Z}_1^k\boldsymbol{B}_1)\boldsymbol{Z}_1^k\boldsymbol{B}_1(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(2\boldsymbol{I}_2 - \boldsymbol{Z}_2^k\boldsymbol{B}_2)\boldsymbol{Z}_2^k\boldsymbol{B}_2(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}\\ &\quad= (2\boldsymbol{I}_1 - \boldsymbol{Z}_1^k\boldsymbol{B}_1)\boldsymbol{Z}_1^k\boldsymbol{B}_1(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[\boldsymbol{Z}_2^k\boldsymbol{B}_2(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}(2\boldsymbol{I}_2 - \boldsymbol{Z}_2^k\boldsymbol{B}_2)^{\mathrm{T}}\\ &\quad= \boldsymbol{Z}_1^{k+1}\boldsymbol{Q}(\boldsymbol{Z}_2^{k+1})^{\mathrm{T}}.\end{aligned}$$

For the case of $k + 1$, (11) is true. Then, this completes the proof. □

Theorem 1. *If the weights ω_1 and ω_2 satisfy the condition (4) and the collocation matrices $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are of full column rank, then the iterative tensor product surface sequence generated by the iterative format (7) is convergent to the least-squares fitting result of the initial data points $\boldsymbol{Q}$.*

Proof. Applying Lemma 1, $\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1$ and $\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2$ are both non-singular. According to the iterative format (7), we have

$$\begin{aligned}&\boldsymbol{P}^{k+1} - (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}\\ &\quad= \boldsymbol{P}^k + \boldsymbol{Z}_1^{k+1}(\boldsymbol{Q} - \boldsymbol{B}_1\boldsymbol{P}^k\boldsymbol{B}_2^{\mathrm{T}})(\boldsymbol{Z}_2^{k+1})^{\mathrm{T}} - (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}.\end{aligned}$$

Using Lemma 2, substituting (11) into the abovementioned equation, it yields

$$\begin{aligned}
&\boldsymbol{P}^{k+1} - (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{T}]^{\mathrm{T}} \\
&\quad = \boldsymbol{P}^{k} - \boldsymbol{Z}_1^{k+1}\boldsymbol{B}_1\boldsymbol{P}^{k}\boldsymbol{B}_2^{\mathrm{T}}(\boldsymbol{Z}_2^{k+1})^{\mathrm{T}} - (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}} \\
&\qquad + \boldsymbol{Z}_1^{k+1}\boldsymbol{B}_1(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}(\boldsymbol{Z}_2^{k+1}\boldsymbol{B}_2)^{\mathrm{T}} \\
&\quad = [\boldsymbol{P}^{k} - (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}((\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}})^{\mathrm{T}}] \\
&\qquad - \boldsymbol{Z}_1^{k+1}\boldsymbol{B}_1\left\{\boldsymbol{P}^{k} - (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}\right\}(\boldsymbol{Z}_2^{k+1}\boldsymbol{B}_2)^{\mathrm{T}}
\end{aligned} \tag{12}$$

The above equation is expressed in matrix form. For the convenience of analysis, we need to rearrange it in column-vector form. For example, the initial data point $\boldsymbol{Q}$ is expressed as

$$\overrightarrow{\boldsymbol{Q}} = [Q_{00}, Q_{01}, \cdots, Q_{0m_2}, Q_{10}, \cdots, Q_{1m_2}, \cdots, Q_{m_1m_2}]^{\mathrm{T}}.$$

$\boldsymbol{P}$ is treated analogously. We define

$$\boldsymbol{P}^{\infty} = (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}. \tag{13}$$

Applying the properties of the Kronecker product [30], (12) is equivalent to

$$\begin{aligned}
\overrightarrow{\boldsymbol{P}^{k+1} - \boldsymbol{P}^{\infty}} &= \overrightarrow{\boldsymbol{P}^{k} - \boldsymbol{P}^{\infty}} - (\boldsymbol{Z}_1^{k+1}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{k+1}\boldsymbol{B}_2)\, \overrightarrow{\boldsymbol{P}^{k} - \boldsymbol{P}^{\infty}} \\
&= [\boldsymbol{I} - (\boldsymbol{Z}_1^{k+1}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{k+1}\boldsymbol{B}_2)]\, \overrightarrow{\boldsymbol{P}^{k} - \boldsymbol{P}^{\infty}}.
\end{aligned}$$

The above iterative process is repeated, and we obtain

$$\begin{aligned}
\overrightarrow{\boldsymbol{P}^{k+1} - \boldsymbol{P}^{\infty}} &= [\boldsymbol{I} - (\boldsymbol{Z}_1^{k+1}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{k+1}\boldsymbol{B}_2)][\boldsymbol{I} - (\boldsymbol{Z}_1^{k}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{k}\boldsymbol{B}_2)]\overrightarrow{\boldsymbol{P}^{k+1} - \boldsymbol{P}^{\infty}} \\
&= \cdots \\
&= \prod_{s=1}^{k+1}[\boldsymbol{I} - (\boldsymbol{Z}_1^{s}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{s}\boldsymbol{B}_2)]\, \overrightarrow{\boldsymbol{P}^{0} - \boldsymbol{P}^{\infty}}.
\end{aligned} \tag{14}$$

where $(\boldsymbol{Z}_1^{s}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{s}\boldsymbol{B}_2)$ is the Kronecker product of $\boldsymbol{Z}_1^{s}\boldsymbol{B}_1$ and $\boldsymbol{Z}_2^{s}\boldsymbol{B}_2$, and $\boldsymbol{I}$ is the identity matrix of order $(n_1+1)(n_2+1)$.

According to the iterative process of $\boldsymbol{Z}_1^{s}$ and $\boldsymbol{Z}_2^{s}$ shown in (7), we have

$$\rho(\boldsymbol{Z}_1^{1}) = (2 - \omega_1\lambda_i)\omega_1\lambda_i,\ \rho(\boldsymbol{Z}_2^{1}) = (2 - \omega_2\mu_j)\omega_2\mu_j.$$

When the weights ω_1 and ω_2 satisfy the condition (4), i.e.,

$$0 < \omega_1 < 2\lambda_0^{-1},\ \omega_1 \neq \lambda_i^{-1},\ 0 < \omega_2 < 2\mu_0^{-1},\ \omega_2 \neq \mu_j^{-1},$$

there is

$$-1 < -(1 - \omega_1\lambda_i)^2, -(1 - \omega_2\mu_j)^2 \leq 0,$$

It means

$$0 < \rho(\boldsymbol{Z}_1^{1}), \rho(\boldsymbol{Z}_2^{1}) \leq 1.$$

Therefore, $\rho(\boldsymbol{I} - (\boldsymbol{Z}_1^{1}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{1}\boldsymbol{B}_2)) = 1 - \rho(\boldsymbol{Z}_1^{1}\boldsymbol{B}_1) \cdot \rho(\boldsymbol{Z}_2^{1}\boldsymbol{B}_2)$ according to [31], and the value of $\rho(\boldsymbol{I} - (\boldsymbol{Z}_1^{1}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{1}\boldsymbol{B}_2))$ belongs to $[0, 1)$.

Suppose there is $0 < \rho(\boldsymbol{Z}_1^{s}), \rho(\boldsymbol{Z}_2^{s}) \leq 1$, $s \geq 1$, then using the iterative process of $\boldsymbol{Z}_1^{s}$ and $\boldsymbol{Z}_2^{s}$ shown in (7), we obtain

$$0 < \rho(\boldsymbol{Z}_1^{s+1}\boldsymbol{B}_1) = 1 - [\rho(\boldsymbol{Z}_1^{s}\boldsymbol{B}_1) - 1]^2 \leq 1,$$

and

$$0 < \rho(\boldsymbol{Z}_2^{s+1}\boldsymbol{B}_2) = 1 - [\rho(\boldsymbol{Z}_2^{s}\boldsymbol{B}_2) - 1]^2 \leq 1.$$

The assumption is true for the case of $s+1$. Using mathematical induction, we have

$$0 < \rho(\boldsymbol{Z}_1^{s}), \rho(\boldsymbol{Z}_2^{s}) \leq 1,\ s \geq 1$$

Further, we have

$$\rho[\boldsymbol{I} - (\boldsymbol{Z}_1^{s}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{s}\boldsymbol{B}_2)] = 1 - \rho(\boldsymbol{Z}_1^{s}\boldsymbol{B}_1) \cdot \rho(\boldsymbol{Z}_2^{s}\boldsymbol{B}_2) \in [0,1),$$

and the spectral radius of the iterative matrix $\prod_{s=1}^{k+1}[\boldsymbol{I} - (\boldsymbol{Z}_1^{s}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{s}\boldsymbol{B}_2)]$ has the value range $[0,1)$. Therefore,

$$\lim_{k\to\infty}\prod_{s=1}^{k+1}[\boldsymbol{I} - (\boldsymbol{Z}_1^{s}\boldsymbol{B}_1) \otimes (\boldsymbol{Z}_2^{s}\boldsymbol{B}_2)] = 0_{(n_1+1)(n_2+1)},$$

where $\mathbf{0}_{(n_1+1)(n_2+1)}$ is a zero matrix of order $(n_1+1)(n_2+1)$.

Using (14), when $k \to \infty$, it follows

$$\boldsymbol{P}^{k+1} \overrightarrow{-}\ \boldsymbol{P}^{\infty} = \mathbf{0}_{(n_1+1)(n_2+1)\times 2}.$$

We rewrite the above equation in matrix form and replace $\boldsymbol{P}^{\infty}$ by (13). It is equivalent to $\boldsymbol{P}^{k} \to (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)^{-1}\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{Q}[(\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2)^{-1}\boldsymbol{B}_2^{\mathrm{T}}]^{\mathrm{T}}$ as $k \to \infty$, which means the surface sequence $\left\{\boldsymbol{P}^{k}(u,v), k = 0,1,...\right\}$ (8) obtained by the iterative format (7) is convergent, and the limit surface is the least-squares approximation surface of the initial data points $\left\{\boldsymbol{Q}_{i,j}\right\}_{i=0,j=0}^{m_1,m_2}$. This completes the proof. □

4. Implementation

This section introduces the preparation for the fitting algorithm.

4.1. Parameterization of the Initial Data Points

Given a data point set $\left\{\boldsymbol{Q}_{i,j}\right\}_{i=0,j=0}^{m_1,m_2}$, each point $\boldsymbol{Q}_{i,j}$ is assigned to a pair of parameter values (u_i, v_j), where u_i is defined as the row parameter of $\boldsymbol{Q}_{i,j}$ in the u -direction, and v_j is defined as the column parameter of $\boldsymbol{Q}_{i,j}$ in the v-direction (See Figure 1a). The procedure for calculating (u_i, v_j) is illustrated in Figure 1.

First, we calculate the parameter $t_{i,j}$ for $\boldsymbol{Q}_{i,j}$ using the normalized accumulated chord parameterization method [32], i.e.,

$$t_{0,j} = 0,\ t_{i,j} = t_{i-1,j} + \frac{\|\boldsymbol{Q}_{i,j} - \boldsymbol{Q}_{i-1,j}\|}{\sum_{i=1}^{m_1}\|\boldsymbol{Q}_{i,j} - \boldsymbol{Q}_{i-1,j}\|},\ i = 1,2,\cdots,m_1,\ j = 0,1,\cdots,m_2.$$

Then, the row parameter u_i of point $\boldsymbol{Q}_{i,j}$ in the i-th row is defined by the average value of the parameters $t_{i,j}$ of the data points in the i-th row (See Figure 1b), i.e.,

$$u_i = \frac{1}{m_2+1}\sum_{j=0}^{m_2} t_{i,j}, \quad i = 0,1,\cdots,m_1.$$

The column parameter v_j of $\boldsymbol{Q}_{i,j}$ can be calculated in the same way. Therefore, the parameter pair (u_i, v_j) of each point $\boldsymbol{Q}_{i,j}$ can be derived.

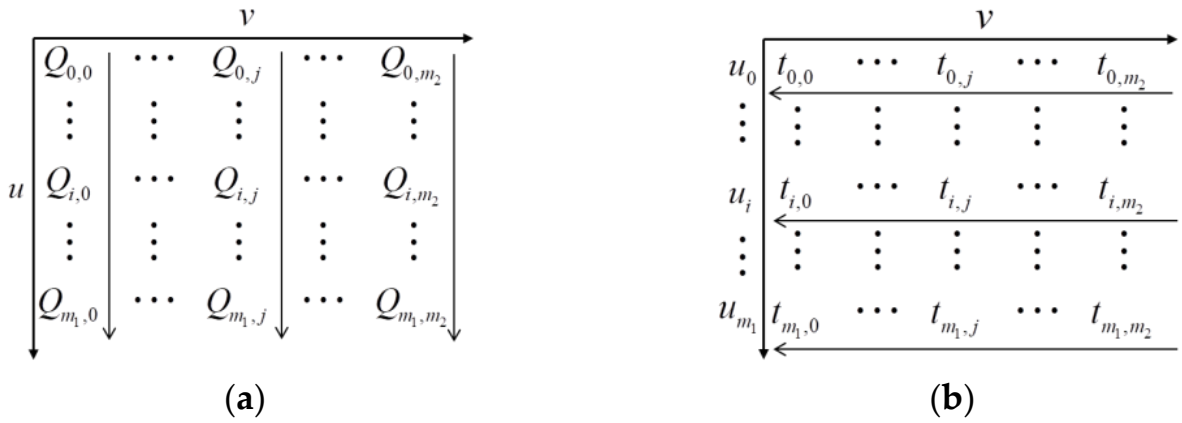


Figure 1. (**a**) u-direction and the v-direction, (**b**) row parameter u_i.

4.2. Selection of the Initial Control Points

According to Theorem 1, the iterative surface sequence converges to the least-squares fitting result of the data points, no matter how we choose the initial control points. As a practical matter, the control points cannot be selected completely at random. Considering that the shape of the data set is mainly characterized by points with a large curvature, we choose the points with a larger curvature as the initial control points to make the proposed method converge quickly.

Similar to the parameterization procedure in Section 4.1, given a data point set $\left\{\boldsymbol{Q}_{i,j}\right\}_{i=0,j=0}^{m_1,m_2}$, each point $\boldsymbol{Q}_{i,j}$ is assigned to a curvature pair (k_i^u, k_j^v). First, the total curvature $k_{i,j}$ is calculated for each point $\boldsymbol{Q}_{i,j}$ as follows.

According to the formula of discrete curvature [33], the discrete curvature $k_{i,j}^u$ in the u-direction is expressed as

$$k_{i,j}^u = \frac{\|\Delta \boldsymbol{Q}_{i,j} \times \Delta^2 \boldsymbol{Q}_{i,j}\|}{\|\Delta \boldsymbol{Q}_{i,j}\|^3}, i = 0, 1, \cdots, m_1,$$

where $\Delta \boldsymbol{Q}_{i,j}$ and $\Delta^2 \boldsymbol{Q}_{i,j}$ are the first-order and second-order differences of $\boldsymbol{Q}_{i,j}$ in the u-direction, i.e.,

$$\begin{aligned}
\Delta \boldsymbol{Q}_{0,j} &= \frac{\boldsymbol{Q}_{1,j} - \boldsymbol{Q}_{0,j}}{\|\boldsymbol{Q}_{1,j} - \boldsymbol{Q}_{0,j}\|}, \\
\Delta \boldsymbol{Q}_{i,j} &= \frac{\boldsymbol{Q}_{i+1,j} - \boldsymbol{Q}_{i-1,j}}{\|\boldsymbol{Q}_{i+1,j} - \boldsymbol{Q}_{i,j}\| + \|\boldsymbol{Q}_{i,j} - \boldsymbol{Q}_{i-1,j}\|}, \ i = 1, 2, \cdots, m_1 - 1, \\
\Delta \boldsymbol{Q}_{m_1,j} &= \frac{\boldsymbol{Q}_{m_1,j} - \boldsymbol{Q}_{m_1-1,j}}{\|\boldsymbol{Q}_{m_1,j} - \boldsymbol{Q}_{m_1-1,j}\|}, \\
\Delta^2 \boldsymbol{Q}_{0,j} &= \frac{\Delta \boldsymbol{Q}_{1,j} - \Delta \boldsymbol{Q}_{0,j}}{\|\boldsymbol{Q}_{1,j} - \boldsymbol{Q}_{0,j}\|}, \\
\Delta^2 \boldsymbol{Q}_{i,j} &= \frac{\Delta \boldsymbol{Q}_{i+1,j} - \Delta \boldsymbol{Q}_{i-1,j}}{\|\boldsymbol{Q}_{i+1,j} - \boldsymbol{Q}_{i,j}\| + \|\boldsymbol{Q}_{i,j} - \boldsymbol{Q}_{i-1,j}\|}, \ i = 1, 2, \cdots, m_1 - 1, \\
\Delta^2 \boldsymbol{Q}_{m_1,j} &= \frac{\Delta \boldsymbol{Q}_{m_1,j} - \Delta \boldsymbol{Q}_{m_1-1,j}}{\|\boldsymbol{Q}_{m_1,j} - \boldsymbol{Q}_{m_1-1,j}\|}.
\end{aligned}$$

The discrete curvature $k_{i,j}^v$ of $\boldsymbol{Q}_{i,j}$ in the v-direction is treated analogously. Therefore, the total curvature $k_{i,j}$ of $\boldsymbol{Q}_{i,j}$ is calculated by

$$k_{i,j} = \left(k_{i,j}^u\right)^2 + \left(k_{i,j}^v\right)^2, \ i = 0, 1, \cdots, m_1, \ j = 0, 1, \cdots, m_2.$$

Next, the row curvature k_i^u of $\boldsymbol{Q}_{i,j}$ is obtained by averaging the total curvatures in the v-direction

$$k_i^u = \frac{1}{m_2+1}\sum_{j=0}^{m_2} k_{i,j},\ i = 0, 1, \cdots, m_1.$$

The column curvature k_j^v of $\boldsymbol{Q}_{i,j}$ is calculated in a similar way. Therefore, the curvature pair (k_i^u, k_j^v) of each data point $\boldsymbol{Q}_{i,j}$ is derived.

To choose the control points $\{\boldsymbol{P}_{h,l}\}_{h=0,l=0}^{n_1,n_2}(n_1 < m_1, n_2 < m_2)$ of the initial iterative surface from the original data points $\left\{\boldsymbol{Q}_{i,j}\right\}_{i=0,j=0}^{m_1,m_2}$, we just select their subscripts (row subscript and column subscript) based on their curvature pairs. Since the boundary points are also the feature points, four corners are selected as the initial control points, namely, the row subscripts $0, m_1$ and the column subscripts $0, m_2$. For the row curvature k_i^u, choose the row subscripts $\{i_1, ..., i_{n_1-1}\}$ from $\{1, ..., m_1 - 1\}$, whose row curvatures are top $n_1 - 1$, where $n_1 < m_1$. For the column curvature k_j^v, choose the column subscripts $\{j_1, ..., j_{n_2-1}\}$ from $\{1, ..., m_2 - 1\}$, whose column curvatures are top $n_2 - 1$, where $n_2 < m_2$. Therefore, the subscripts of the initial control points belong to the following subscript sequence

$$\{i_0 = 0, i_1, \cdots, i_{n_1} = m_1\} \times \{j_0 = 0, j_1, \cdots, j_{n_2} = m_2\},$$

and the initial position of the control points can be determined while the parameter pair (u_i, v_j) of each control point satisfies

$$(u_i, v_j) \in \left\{u_{i_0}, u_{i_1}, \cdots, u_{i_{n_1}}\right\} \times \left\{v_{j_0}, v_{j_1}, \cdots, v_{j_{n_2}}\right\}.$$

4.3. Knot Vectors and Fitting Error

The knot vectors for a bi-cubic tensor product B-spline surface are defined by the following technique of averaging [32]

$$\begin{gathered}
\overline{u}_0 = \overline{u}_1 = \overline{u}_2 = \overline{u}_3 = u_0, \overline{u}_{n_1} = \overline{u}_{n_1+1} = \overline{u}_{n_1+2} = \overline{u}_{n_1+3} = u_{n_1}, \\
\overline{u}_{k_u+3} = \frac{1}{3}\sum_{k=p}^{p+2} u_{i_k}, p = 1, 2, \cdots, n_1 - 3, \\
\overline{v}_0 = \overline{v}_1 = \overline{v}_2 = \overline{v}_3 = v_0, \overline{v}_{n_2} = \overline{v}_{n_2+1} = \overline{v}_{n_2+2} = \overline{v}_{n_2+3} = v_{n_2}, \\
\overline{v}_{k_v+3} = \frac{1}{3}\sum_{k=q}^{q+2} v_{j_k}, q = 1, 2, \cdots, n_2 - 3.
\end{gathered}$$

The fitting error E_k of the k-th iteration is defined by

$$E_k = \sum_{i=0}^{m_1}\sum_{j=0}^{m_2}\left\|\boldsymbol{Q}_{i,j} - \sum_{h=0}^{n_1}\sum_{l=0}^{n_2}\boldsymbol{B}_h(u_i)\boldsymbol{B}_l(v_j)\boldsymbol{P}_{h,l}^k\right\|^2 \tag{15}$$

4.4. Algorithm Implementation

The process of fitting a data set with the proposed method is described in Algorithm 1.

Algorithm 1. Surface fitting of the data points by the accelerated LSPIA method

Input: The data point set $\left\{\boldsymbol{Q}_{i,j}\right\}_{i=0,j=0}^{m_1,m_2}$ and a given iterative tolerance ε.
Output: The fitting surface $\boldsymbol{P}^k(u,v)$.
Step 1: Calculate the parameter pair $\left(u_i,v_j\right)$ for each point $\boldsymbol{Q}_{i,j}$ according to Section 4.1.
Step 2: Choose the initial control points $\boldsymbol{P}^0$ according to Section 4.2.
Step 3: Calculate the knot vectors according to Section 4.3.
Step 4: Compute $\boldsymbol{Z}_1^0$ and $\boldsymbol{Z}_2^0$ according to (3).
$k = 0$;
do

- Calculate the difference vectors $\boldsymbol{\delta}^k = \boldsymbol{Q} - \boldsymbol{B}_1\boldsymbol{P}^k\boldsymbol{B}_2^{\mathrm{T}}$;
- Calculate $\boldsymbol{Z}_1^{k+1} = (2\boldsymbol{I}_1 - \boldsymbol{Z}_1^k\boldsymbol{B}_1)\boldsymbol{Z}_1^k$ and $\boldsymbol{Z}_2^{k+1} = (2\boldsymbol{I}_2 - \boldsymbol{Z}_2^k\boldsymbol{B}_2)\boldsymbol{Z}_2^k$;
- Calculate the adjustment vectors $\Delta^k = \boldsymbol{Z}_1^{k+1}\boldsymbol{\delta}^k\left(\boldsymbol{Z}_2^{k+1}\right)^{\mathrm{T}}$;
- Construct the control points $\boldsymbol{P}^{k+1} = \boldsymbol{P}^k + \Delta^k$ for the $(k+1)$-*st* iteration;
- Calculate the fitting error E_k by (15);
- $k = k + 1$;

While $(|E_{k+1} - E_k| < \varepsilon)$.

5. Numerical Examples

In this section, we present five examples and compare the results with those of the traditional LSPIA method [18] to show the efficiency of the proposed LSPIA algorithm for the surface fitting of data set. The fitting surface had bi-cubic B-spline basis functions, since the bi-cubic tensor product B-spline surface is simple and widely used in CAD/CAM. It is hard to accurately calculate the optimal weights; so, we defined ω_1, ω_2 in the same way as the weight in [18] for practical applications, i.e.,

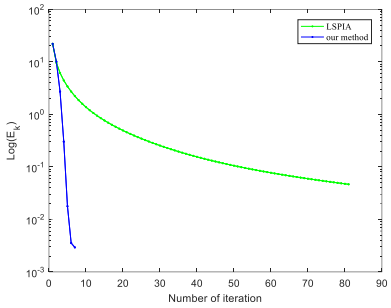
$$\omega_1 = 2\|\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1\|_\infty^{-1},\ \omega_2 = 2\|\boldsymbol{B}_2^{\mathrm{T}}\boldsymbol{B}_2\|_\infty^{-1}.$$

The point sets in the five examples were as follows.

Example 1: 121×161 grid data points sampled from a human face collection.
Example 2: 21×21 grid data points sampled from an ear shape flake collection.
Example 3: 41×61 grid data points sampled from a mold collection.
Example 4: 81×61 grid data points sampled from a tooth shape flake collection.
Example 5: 501×501 grid data points sampled from a peaks (501) function collection.

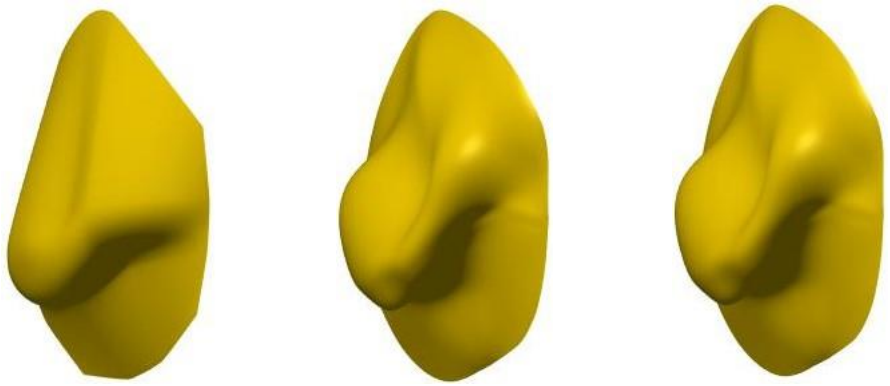
The fitting surfaces and the fitting errors are plotted in Figures 2–6, where the point grids and the initial iterative surface are shown in (a), the iterative surfaces after five iterations are provided in (b), the limit fitting surfaces are shown in (c), and the comparison chart of the fitting error vs. the number of iterations for the proposed method and the traditional LSPIA method are presented in (d). In (d), it is easy to observe that the fitting error of our proposed method decreased sharply and satisfied the termination criterion quickly.

(**a**) Step 0. (**b**) Step 5. (**c**) LS fitting surface.

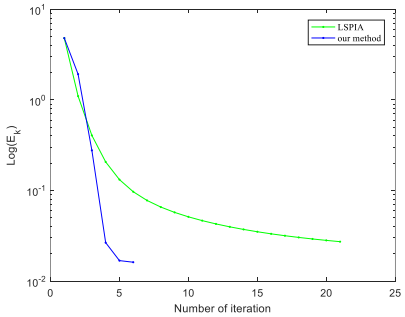


(**d**) Error vs. iteration.

Figure 2. A data point grid with 121 × 161 points was fitted by a bi−cubic tensor product B−spline surface with 60 × 80 control points.

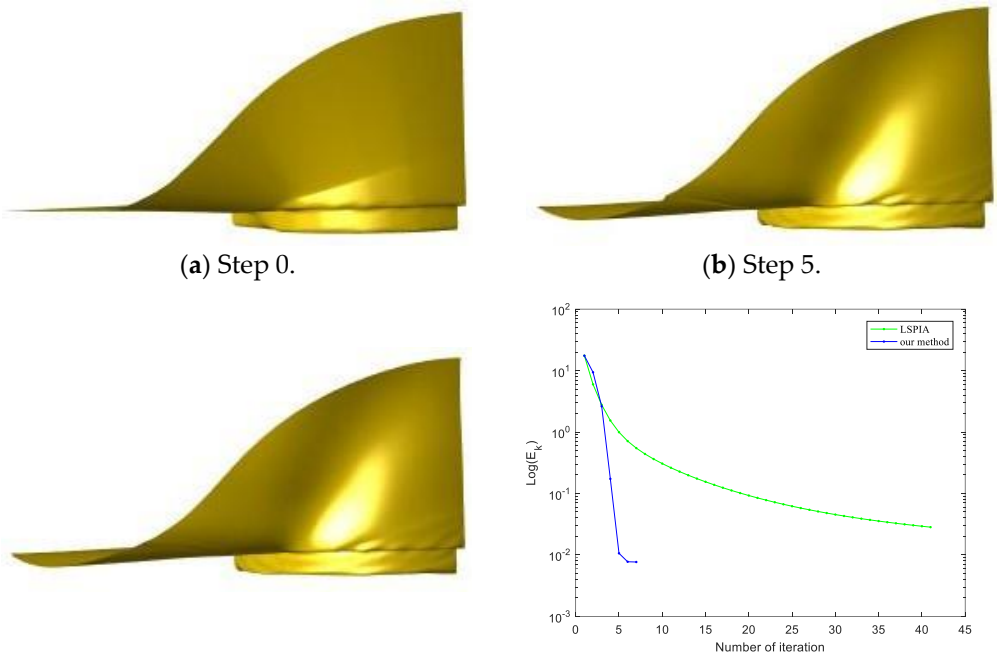


(**a**) Step 0. (**b**) Step 5. (**c**) LS fitting surface.



(**d**) Error vs. iteration.

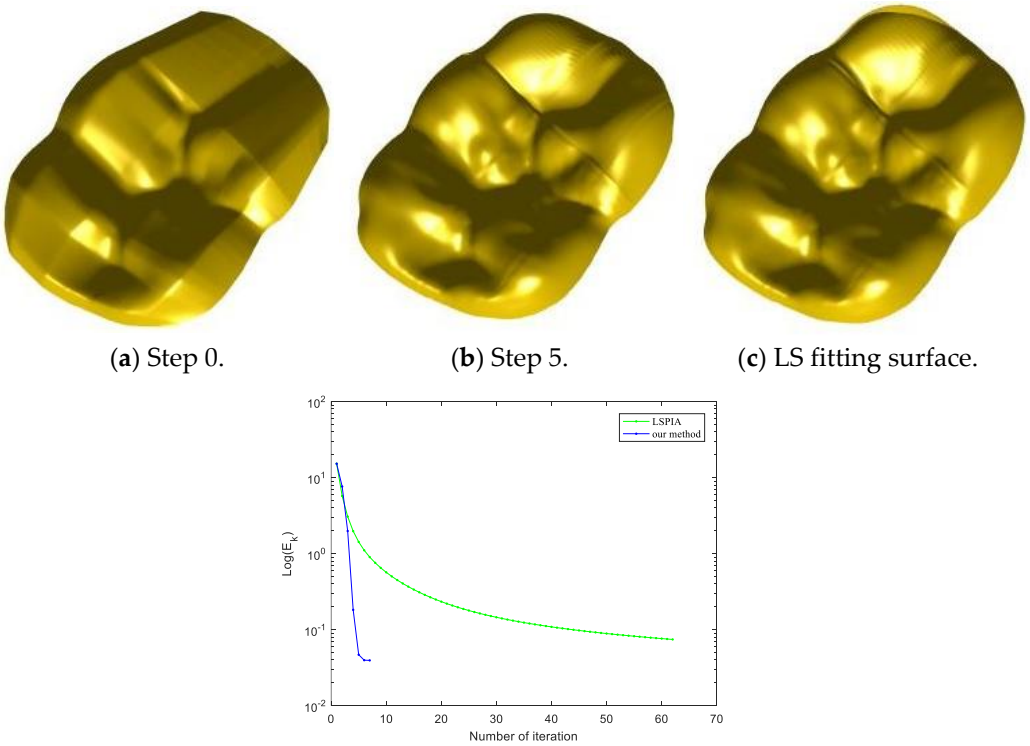
Figure 3. A data point grid with 21 × 21 points was fitted by a bi−cubic tensor product B−spline surface with 10 × 10 control points.



(**a**) Step 0. (**b**) Step 5.

(**c**) LS fitting surface. (**d**) Error vs. iteration.

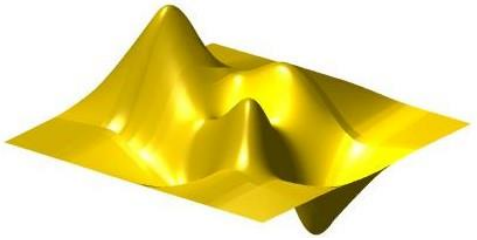
Figure 4. A data point grid with 41 × 61 points was fitted by a bi−cubic tensor product B−spline surface with 20 × 30 control points.



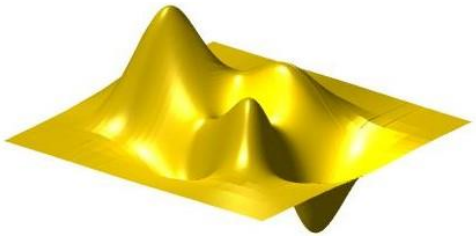
(**a**) Step 0. (**b**) Step 5. (**c**) LS fitting surface.

(**d**) Error vs. iteration.

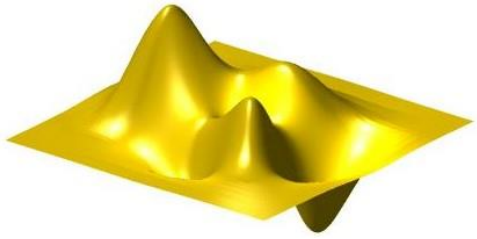
Figure 5. A data point grid with 81 × 61 points was fitted by a bi−cubic tensor product B−spline surface with 40 × 30 control points.



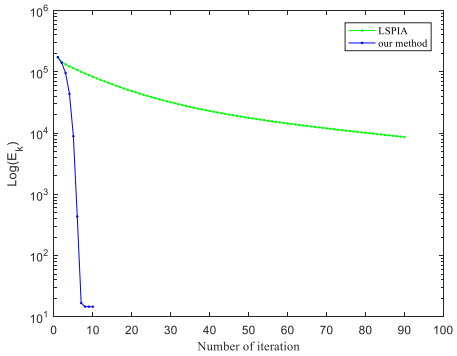
(**a**) Step 0.



(**b**) Step 5.



(**c**) LS fitting surface.



(**d**) Error vs. iteration.

Figure 6. A data point grid with 501 × 501 points was fitted by a bi−cubic tensor product B−spline surface with 250 × 250 control points.

Table 1 contains the fitting errors E_k (k = 0, 1,..., 6), E_{stop}, and E_∞ for examples 1–5 when using our method and the traditional LSPIA method. The termination criterion is $|E_{k+1} - E_k| < 10^{-3}$. E_{stop} is the fitting error when the algorithm is stopped, E_∞ represents the error between $\boldsymbol{Q}$ and $\boldsymbol{P}^\infty$. If E_k is null in Table 1, it means that the termination criterion is satisfied in the previous iteration. In Table 1, we can see that the error of the proposed method was less than that of the LSPIA method after the same iterations, and the proposed algorithm was terminated after about five or six iterations in examples 1–4.

Table 1. (k = 0, 1,..., 6), E_{stop} and E_∞ for examples 1–5.

Examples	Methods	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_{stop}	E_∞
Ex. 1	LSPIA	21.96525	9.58174	6.13142	4.42516	3.39926	2.71392	2.22640	0.04658	0.00292
	Ours	21.96525	10.07568	2.70966	0.30180	0.01780	0.00355	0.00293	0.00293	
Ex. 2	LSPIA	4.80661	1.10115	0.40587	0.20703	0.13203	0.09708	0.07782	0.02730	0.01619
	Ours	4.80661	1.92369	0.27769	0.02663	0.01687	0.01620			
Ex. 3	LSPIA	17.46394	5.96253	2.76524	1.53887	0.99237	0.71154	0.54684	0.02828	0.00766
	Ours	17.46394	9.37228	2.61701	0.17271	0.01061	0.00773	0.00766		
Ex. 4	LSPIA	15.25581	5.74124	3.07872	1.97844	1.42757	1.10820	0.90186	0.07447	0.03950
	Ours	15.25581	7.64569	1.98735	0.18241	0.04691	0.03974	0.03950		
Ex. 5	LSPIA	173,996.46	144,906.89	132,412.83	122,822.74	114,511.11	107,057.94	100,288.95	15.51962	14.63951
	Ours	173,996.46	140,489.44	95,713.553	43,647.269	8899.5952	434.91927	16.787049	14.63951	

The error between the control point $\boldsymbol{P}^k$ after k iterations and the limit control point $\boldsymbol{P}^\infty$ is defined by

$$e_k = \sum_{h=0}^{n_1} \sum_{l=0}^{n_2} \|\boldsymbol{P}_{h,l}^k - \boldsymbol{P}_{h,l}^\infty\|^2.$$

Table 2 shows that the error e_k of the proposed method was less than that of the LSPIA method after k iterations, and the control point $\boldsymbol{P}^k$ was much closer to the limit control point.

Table 2. (k = 0, 1,..., 7) for examples 1–5.

Examples	Methods	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7
Ex. 1	LSPIA	4.496×10^{0}	3.406×10^{0}	2.840×10^{0}	2.454×10^{0}	2.167×10^{0}	1.943×10^{0}	1.763×10^{0}	1.615×10^{0}
	Ours	4.496×10^{0}	2.558×10^{0}	1.024×10^{0}	2.497×10^{-1}	3.112×10^{-2}	1.891×10^{-3}	5.310×10^{-5}	1.163×10^{-7}
Ex. 2	LSPIA	2.107×10^{0}	9.672×10^{-1}	6.391×10^{-1}	4.975×10^{-1}	4.204×10^{-1}	3.710×10^{-1}	3.354×10^{-1}	3.079×10^{-1}
	Our	2.107×10^{0}	9.146×10^{-1}	2.761×10^{-1}	8.403×10^{-2}	1.375×10^{-2}	3.041×10^{-4}	7.247×10^{-8}	1.754×10^{-15}
Ex. 3	LSPIA	2.419×10^{0}	1.406×10^{0}	1.013×10^{0}	7.989×10^{-1}	6.639×10^{-1}	5.694×10^{-1}	4.986×10^{-1}	4.430×10^{-1}
	Ours	2.419×10^{0}	1.165×10^{0}	3.351×10^{-1}	6.330×10^{-2}	8.403×10^{-3}	3.900×10^{-4}	3.268×10^{-6}	4.527×10^{-10}
Ex. 4	LSPIA	3.026×10^{0}	2.054×10^{0}	1.632×10^{0}	1.383×10^{0}	1.215×10^{0}	1.091×10^{0}	9.940×10^{-1}	9.148×10^{-1}
	Ours	3.026×10^{0}	1.611×10^{0}	6.031×10^{-1}	1.610×10^{-1}	2.331×10^{-2}	1.032×10^{-3}	8.683×10^{-6}	2.193×10^{-9}
Ex. 5	LSPIA	1.159×10^{4}	1.113×10^{4}	1.072×10^{4}	1.035×10^{4}	1.000×10^{4}	9.676×10^{3}	9.372×10^{3}	9.088×10^{3}
	Ours	1.159×10^{4}	1.043×10^{4}	7.890×10^{3}	4.187×10^{3}	1.225×10^{3}	1.295×10^{2}	3.505×10^{0}	9.999×10^{-2}

Table 3 lists the running time and iteration number of the proposed method and the LSPIA method. It can be observed that the iteration number of the LSPIA method was much larger than that of the proposed method. When the termination condition was $|E_{k+1} - E_k| < 10^{-7}$, the iteration number of the LSPIA method was 70–140 times that of our method. The iteration number rapidly increased as the termination criterion decreased. The reason is that our method is a non-stationary iterative method. As the iteration number increased, the spectral radius of the iterative matrix decreased rapidly and tended to zero. Although our iterative format is complex, the running time of the proposed method was less than that of the LSPIA method. Especially, it is obvious that the proposed method is more advantageous and efficient than the LSPIA method, when there is a large-scale fitting problem, such as in Example 5.

Table 3. Comparison of the iteration number and running time of our method and the LSPIA method under the different termination criteria.

Examples	Criterion	LSPIA		Ours	
		Time (s)	Iterations	Time (s)	Iterations
Ex. 1	1×10^{-3}	0.4640	80	0.3863	6
	1×10^{-5}	0.7520	350	0.3971	7
	1×10^{-7}	1.2650	771	0.4210	8
Ex. 2	1×10^{-3}	0.0123	20	0.0102	5
	1×10^{-5}	0.0145	110	0.0118	6
	1×10^{-7}	0.0346	712	0.0128	7
Ex. 3	1×10^{-3}	0.0544	40	0.0512	6
	1×10^{-5}	0.0720	190	0.0526	7
	1×10^{-7}	0.1250	571	0.0537	8
Ex. 4	1×10^{-3}	0.1180	61	0.0951	6
	1×10^{-5}	0.1710	262	0.0962	7
	1×10^{-7}	0.2870	652	0.0971	8
Ex. 5	1×10^{-3}	230.465	4076	7.079	9
	1×10^{-5}	533.765	9154	7.386	10
	1×10^{-7}	1151.947	18,837	8.104	11

6. Conclusions

In this paper, we propose an efficient least-squares progressive iterative method for tensor product surfaces. Combined with the Schulz iterative method for computing the Moore–Penrose inverse, we designed an accelerated LSPIA iterative format with fast convergence rate. In addition, the iterative format was expressed in matrix form, which is different from previous methods. The calculation amount was greatly reduced, since the method avoided the Kronecker product. The presented method is perfectly suitable for

big data fitting in engineering applications. Future research may focus on extending the method to different kinds of PIA methods.

Author Contributions: Conceptualization, Q.H.; Methodology, Q.H.; Software, Z.W.; Validation, R.L.; Writing—original draft, Z.W.; Writing—review & editing, Q.H. and R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 62272406).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated or analyzed in this study are available from the corresponding author on reasonable request.

Acknowledgments: The authors would like to thank the anonymous referees for their valuable suggestions and comments.

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

References

1. Lin, H.; Jin, S.; Liao, H.; Jian, Q. Quality guaranteed all-hex mesh generation by a constrained volume iterative fitting algorithm. *Comput. Aided Des.* **2015**, *67*, 107–117. [CrossRef]
2. Martin, T.; Cohen, E.; Kirby, R. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. *Comput. Aided Des.* **2009**, *26*, 648–664. [CrossRef]
3. Lin, H.; Jin, S.; Hu, Q.; Liu, Z. Constructing B-spline solids from tetrahedral meshes for isogeometric analysis. *Comput. Aided Geom. Des.* **2015**, *35*, 109–120. [CrossRef]
4. Kineri, Y.; Wang, M.; Lin, H.; Maekawa, T. B-spline surface fitting by iterative geometric interpolation/approximation algorithms. *Comput. Aided Des.* **2012**, *44*, 697–708. [CrossRef]
5. Qi, D.; Tian, Z.; Zhang, Y.; Zheng, J.B. The method of numeric polish in curve fitting. *Acta Math. Sin.* **1975**, *18*, 173–184. (In Chinese)
6. De Boor, C. How does Agee's smoothing method work. In Proceedings of the 1979 Army Numerical Analysis and Computers Conference, ARO Report, Madison, WI, USA; 1979; pp. 79–83.
7. Lin, H.; Wang, G.; Dong, C. Constructing iterative non-uniform B-spline curve and surface to fit data points. *Sci. China Ser. Inf. Sci.* **2004**, *47*, 315–331. [CrossRef]
8. Lin, H.W.; Bao, H.J.; Wang, G.J. Totally positive bases and progressive iterative approximation. *Comput. Math. Appl.* **2005**, *50*, 575–586. [CrossRef]
9. Lin, H.; Maekawa, T.; Deng, C. Survey on geometric iteartive methods and their applications. *Comput. Aided Des.* **2018**, *95*, 40–51. [CrossRef]
10. Lin, H. Local progressive-iterative approximation format for blending curves and patches. *Comput. Aided Geom. Des.* **2010**, *27*, 322–339. [CrossRef]
11. Hu, Q.; Wang, J.; Liang, R. Weighted local progressive-iterative approximation property for triangular Bézier surfaces. *Vis. Comput.* **2022**, *28*, 2819–3830. [CrossRef]
12. Lu, L. Weighted progressive iteration approximation and convergence analysis. *Comput. Aided Geom. Des.* **2010**, *27*, 129–137. [CrossRef]
13. Liu, C.; Han, X.; Li, J. The Chebyshev accelerating method for progressive iterative approximation. *Commun. Inf. Syst.* **2017**, *17*, 25–43. [CrossRef]
14. Liu, C.; Han, X.; Li, J. Preconditioned progressive iterative approximation for triangular Bézier patches and its application. *J. Comput. Appl. Math.* **2020**, *366*, 112389. [CrossRef]
15. Ebrahimi, A.; Loghmani, G. A composite iterative procedure with fast convergence rate for the progressive-iteration approximation of curves. *J. Comput. Appl. Math.* **2019**, *359*, 1–15. [CrossRef]
16. Delgado, J.; Peña, J.M. Progressive iterative approximation and bases with the fastest convergence rates. *Comput. Aided Geom. Des.* **2007**, *24*, 10–18. [CrossRef]
17. Hamza, Y.; Lin, H.; Li, Z. Implicit progressive-iterative approximation for curve and surface reconstruction. *Comput. Aided Geom. Des.* **2020**, *77*, 101817. [CrossRef]
18. Deng, C.; Lin, H. Progressive and iterative approximation for least squares B-spline curve and surface fitting. *Comput. Aided Des.* **2014**, *47*, 32–44. [CrossRef]
19. Wang, H. On extended progressive and iterative approximation for least squares fitting. *Vis. Comput.* **2022**, *38*, 591–602. [CrossRef]
20. Hu, Q.; Wang, J.; Wang, G. Improved least square progressive iterative approximation format for triangular Bezier surfaces. *J. Comput. Aided Des. Comput. Graph.* **2022**, *34*, 777–783.
21. Sajavičius, S. Hyperpower least squares progressive iterative approximation. *J. Comput. Appl. Math.* **2023**, *422*, 114888. [CrossRef]

22. Lin, H.; Cao, Q.; Zhang, X. The convergence of least-squares progressive iterative approximation for singular least-squares fitting system. *J. Syst. Sci. Complex.* **2018**, *31*, 1618–1632. [CrossRef]
23. Shi, F. *Computer Aided Geometric Design with Non-Uniform Rational B-Splines*; Higher Education Press: Beijing, China, 2013. (In Chinese)
24. Massarwi, F.; van Sosin, B.; Elber, G. Untrimming: Precise conversion of trimmed-surfaces to tensor-product surfaces. *Comput. Graph.* **2018**, *70*, 80–91. [CrossRef]
25. Vaitkus, M.; Várady, T. Parameterizing and extending trimmed regions for tensor-product surface fitting. *Comput. Aided Des.* **2018**, *104*, 125–140. [CrossRef]
26. Marco, A.; Martínez, J.J. A fast and accurate algorithm for solving Bernstein–Vandermonde linear systems. *Linear Algebra Appl.* **2007**, *422*, 616–628. [CrossRef]
27. Schulz, G. Iterative berechung der reziproken matrix. *ZAMM J. Appl. Math. Mech. Z. Angew. Math. Mech.* **1933**, *13*, 57–59. [CrossRef]
28. De Boor, C. Total positivity of the spline collocation matrix. *Indiana Univ. Math. J.* **1976**, *25*, 541–551. [CrossRef]
29. Penrose, R. A generalized inverse for matrices. In *Mathematical Proceedings of the Cambridge Philosophical Society*; Cambridge University Press: Cambridge, UK, 1955; Volume 51, pp. 406–413.
30. Henderson, H.; Searle, S. The vec-permutation matrix, the vec operator and Kronecker products: A review. *Linear Multilinear Algebra* **1981**, *9*, 271–288. [CrossRef]
31. Brewer, J. Kronecker products and matrix calculus in system theory. *IEEE Trans. Circuits Syst.* **1978**, *25*, 772–781. [CrossRef]
32. Piegl, L.; Tiller, W. *The NURBS Book*; Springer: Berlin/Heidelberg, Germany, 1997.
33. Ray, B.; Pandyan, R. ACORD—An adaptive corner detector for planar curves. *Pattern Recognit.* **2003**, *36*, 703–708. [CrossRef]