mathematics

MDPI

Article

Progressive Iterative Approximation of Non-Uniform Cubic B-Spline Curves and Surfaces via Successive Over-Relaxation Iteration

Huahao Shou [1], Liangchen Hu [2,*] and Shiaofen Fang [3]

[1] College of Science, Zhejiang University of Technology, Hangzhou 310023, China
[2] School of Computer and Information, Anhui Normal University, Wuhu 241002, China
[3] Department of Computer & Information Science, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202, USA
* Correspondence: cslc.hu@ahnu.edu.cn

Abstract: Geometric iteration (GI) is one of the most efficient curve- or surface-fitting techniques in recent years, which is famous for its remarkable geometric significance. In essence, GI can be thought of as the sum of iterative methods for solving systems of linear equations, such as progressive iterative approximation (PIA) which relies on the theory of Richardson iteration. Thus, when the curve- or surface-fitting error is at a desired level, we want to have as few iterations as possible to improve efficiency when dealing with large data sets. Based on the idea of successive over-relaxation (SOR) iteration, we formulate a faster PIA curve and surface interpolation scheme using classical non-uniform cubic B-splines, named SOR-PIA. The genetic algorithm is utilized to estimate the best approximate relaxation factor of SOR-PIA. Similar to standard PIA, SOR-PIA can also be regarded as a process in which the control points move in one direction, but it can greatly reduce the number of iterations in the iterative process with the same fitting accuracy. By comparing with the standard PIA and WPIA algorithms, the effectiveness of the SOR-PIA iterative interpolation algorithm can be verified.

Keywords: progressive iterative approximation; successive over-relaxation iteration; B-spline interpolation

MSC: 65D18; 68U05

check for updates

Citation: Shou, H.; Hu, L.; Fang, S. Progressive Iterative Approximation of Non-Uniform Cubic B-Spline Curves and Surfaces via Successive Over-Relaxation Iteration. *Mathematics* **2022**, *10*, 3766. https://doi.org/10.3390/math10203766

Academic Editors: Juan Cao, Li Zhang and Hongwei Lin

Received: 30 August 2022
Accepted: 9 October 2022
Published: 12 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 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

Data fitting is the technique of constructing mathematical curves or surfaces that best fit a series of data points. Indeed, many novel innovations have emerged in the area of data-fitting techniques, where GI plays a prominent role given the many excellent attributes. GI is a geometrically intuitive method that takes data points as initial control points, adaptively moves along a certain direction through successive iterations, and finally makes the interpolated curve or surface pass through the given data points. GI has a wide field of application with bright prospects, and it is very meaningful to propose a faster iteration strategy for its promotion.

GI originated in the 1970s as a geometrically meaningful iterative method. In 2004, over 20 years later, there was a big breakthrough as Lin derived the profit-and-loss properties in the process of interpolating with non-uniform cubic B-splines [1], and it was proved in 2006 that the properties are also valid for blending the curve and tensor–product blending surface with a normalized totally positive (NTP) basis [2]. The strategy generalized in [2] is known as progressive iterative approximation (PIA), which handles both interpolation and approximation cases, including the extended PIA (EPIA) [3] and the least squares PIA (LSPIA) [4,5]). Later, Carnicer et al. [6] realized that PIA, which had significant geometric implications, was, in fact, essentially the same as Richardson's iteration [7] for a

system of linear equations. In 2007, Maekawa et al. [8] proposed a geometric interpolation technique [8–10], which matches the data points at each iteration of the fitting process with the parameter values at the nearest points to the fitted curve or surface. This differs from Richardson's iteration in that the basis matrix seems to be updated at every iteration. In view of the similarities between PIAs and geometric interpolation methods in the iterative process, they are collectively referred to as the geometric iterative methods (GIMs). Through diversified development, PIA with constraints [11,12] and implicit PIA [13,14] are also widely studied.

To highlight the value of PIA, many researchers have made efforts in accelerating the convergence of PIA and broadening the application of PIA in different fields. For the problem of accelerating PIA, Lu [15,16] derived the optimal weight with the best iteration convergence rate and named it WPIA. In view of the singularity problem caused by possible nodes and knots selection in PIA, an optimal combination scheme was provided in [17]. Indisputably, the PIA approach has been widely extended in many areas. Limit curves can also be generated for position, tangent, and curvature vectors by assigning appropriate geometric constraints [18,19]. By modifying the iterative surface, the position, tangent direction and curvature vector can be interpolated on the isoparametic line at each knot [20]. Aiming at the problem of large-scale data fitting, Lin presented the LSPIA method expressed by T-splines [4]. In the field of finite element analysis [21], GIMs are also utilized to generate hexahedral meshes. This technique restrains the iterative motion of each vertex in the process of generating hexahedral mesh and ensures that the obtained hexahedral mesh is effective. Objectively, the iterative speed of LSPIA is independent of the number of control vertexes, which offers a robust and efficient solution to the tetrahedral mesh equations for generating an ideal NURBS body [22,23]. More broadly, PIAs have also been applied to image/video processing [24–26], hand-drawn curve generation [27], pattern recognition [28], trunk shape modeling [29], and rational curve generation [30]. All of this was outlined in more detail by Lin et al. [31] through two types of models, interpolating and approximating, and they summarized other broad applications of these models in academia and engineering.

In terms of the change of iterative format, a novel iterative curve and surface interpolation algorithm with NTP bases using the Hermitian and skew-Hermitian splitting (HSS) iteration was recently proposed in [32,33]. In contrast, in this paper, we propose another iteration method that also differs from PIA in the iteration form. In Section 2, we show that the PIA and WPIA are essentially designed to solve systems of linear equations using an iterative method. Thus, the matrix iterative format of PIA and WPIA can also be written as a form of $\boldsymbol{P}^k = \boldsymbol{B}\boldsymbol{P}^{k-1} + \boldsymbol{L}$. Since all these iterative methods have the same size of iteration matrix $\boldsymbol{B}$ and constant term matrix $\boldsymbol{L}$ when solving the same problem, we can assume that they consume the same amount of time per iteration. This also means that the method with smaller spectral radii iterates faster if the preparation before iteration is not considered, including the calculation of the matrices $\boldsymbol{B}$ and $\boldsymbol{L}$. In this paper, taking non-uniform cubic B-splines as the fitting tool, we propose a new iterative approximation scheme based on the SOR iteration method commonly used in mathematics.

The remainder of the paper is arranged as follows: using cubic non-uniform B-splines as the fitting tool, we revisit the standard iterative format of PIA in Section 2 and briefly analyze the connections between the standard iterative format and the iterative method of solving the linear equation system. In Section 3, the successive over-relaxation iteration is introduced for fitting the problems of iteratively interpolating curves and surfaces, including the approximate calculation of the best relaxation factor by the genetic algorithm for accelerating the convergence speed. Section 4 describes the reliable numerical experimental results. Section 5 concludes with some summaries and remarks.

2. Standard PIA Format and Its Analysis

In this section, we revisit the standard PIA iteration scheme for curve and surface interpolation with our writing. In essence, it is revealed that the standard PIA is equivalent to solving a system of linear equations.

2.1. Iterative Format of Curve

Given some data points $\{\boldsymbol{Q}_i\}_{i=1}^m$, without loss of generality, we compute the values of the parameters $\{u_i\}_{i=1}^m$ where $u_1 = 0$ and $u_m = 1$ on the initial curve using the accumulative chord length parameterization method. The knot vector $\{0\ 0\ 0\ 0\ u_2\ u_3\ \cdots\ u_{m-1}\ 1\ 1\ 1\ 1\}$ are used to define cubic non-uniform B-spline basis functions, in which the inner knots are directly set as parameter values.

Set $\boldsymbol{P}_i^0 = \boldsymbol{Q}_i, i = 1, \cdots, m$, $\boldsymbol{P}_0^0 = \boldsymbol{P}_1^0$, $\boldsymbol{P}_{m+1}^0 = \boldsymbol{P}_m^0$ and construct the initial iteration curve $\boldsymbol{C}^0(u) = \sum_{i=0}^{m+1} \boldsymbol{P}_i^0 N_{i,3}(u)$. Assuming that the maximum number of iterations is N, the iteration formula can be written as

$$\boldsymbol{P}_i^k = \boldsymbol{P}_i^{k-1} + \boldsymbol{\Delta}_i^{k-1}, i = 1, \cdots, m; k = 1, \cdots, N, \tag{1}$$

where the difference vector can be calculated as

$$\boldsymbol{\Delta}_i^{k-1} = \boldsymbol{Q}_i - \boldsymbol{C}^{k-1}(u_i), i = 1, \cdots, m; k = 1, \cdots, N,$$

and $\boldsymbol{P}_0^k = \boldsymbol{P}_1^k$, $\boldsymbol{P}_{m+1}^k = \boldsymbol{P}_m^k$.

For the cubic non-uniform B-spline with the above-mentioned conditions, there are only three nonzero basis functions; therefore, we have

$$\begin{aligned}\boldsymbol{P}_i^k &= \boldsymbol{P}_i^{k-1} + \boldsymbol{\Delta}_i^{k-1} \\ &= \boldsymbol{P}_i^{k-1} + (\boldsymbol{Q}_i - \boldsymbol{C}^{k-1}(u_i)) \\ &= \boldsymbol{P}_i^{k-1} + (\boldsymbol{Q}_i - \boldsymbol{P}_{i-1}^{k-1} N_{i-1,3}(u_i) - \boldsymbol{P}_i^{k-1} N_{i,3}(u_i) - \boldsymbol{P}_{i+1}^{k-1} N_{i+1,3}(u_i)).\end{aligned}$$

Rewrite the iterative format in Equation (1) to matrix form as follows:

$$\boldsymbol{P}^k = (\boldsymbol{I} - \boldsymbol{N})\boldsymbol{P}^{k-1} + \boldsymbol{Q},$$

where $\boldsymbol{I}$ is an identity matrix,

$$\boldsymbol{N} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ N_{1,3}(u_2) & N_{2,3}(u_2) & N_{3,3}(u_2) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & N_{m-2,3}(u_{m-1}) & N_{m-1,3}(u_{m-1}) & N_{m,3}(u_{m-1}) \\ 0 & \cdots & 0 & 0 & 0 & 0 \quad 1 \end{pmatrix},$$

and $\boldsymbol{P}^k = (\boldsymbol{P}_1^k, \boldsymbol{P}_2^k, \cdots, \boldsymbol{P}_m^k)^T$, $\boldsymbol{Q} = (\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_m)^T$.

2.2. Iterative FORMAT of Surface

Given some data points $\{\boldsymbol{Q}_{i,j}\}_{i=1,j=1}^{m,n}$, without loss of generality, we compute the values of the parameters $\{u_i, v_j\}_{i=1,j=1}^{m,n}$, where $u_1 = v_1 = 0$ and $u_m = v_n = 1$ $\{u_i, v_j\}_{i=1,j=1}^{m,n}$ on the initial surface using the accumulative chord length parameterization method. The knot vectors $\{0\ 0\ 0\ 0\ u_2\ u_3\ \cdots\ u_{m-1}\ 1\ 1\ 1\ 1\}$ and $\{0\ 0\ 0\ 0\ v_2\ v_3\ \cdots\ v_{n-1}\ 1\ 1\ 1\ 1\}$ are used to define bi-cubic non-uniform B-spline basis functions, in which the inner knots are directly set as parameter values.

Set $\boldsymbol{P}_{i,j}^0 = \boldsymbol{Q}_{i,j}$, $\boldsymbol{P}_{i,0}^0 = \boldsymbol{P}_{i,1}^0$, $\boldsymbol{P}_{i,n+1}^0 = \boldsymbol{P}_{i,n}^0$, $\boldsymbol{P}_{0,j}^0 = \boldsymbol{P}_{1,j}^0$, $\boldsymbol{P}_{m+1,j}^0 = \boldsymbol{P}_{m,j}^0$, $i = 1, \cdots, m, j = 1, \cdots, n$; $\boldsymbol{P}_{0,0}^0 = \boldsymbol{P}_{1,1}^0$, $\boldsymbol{P}_{0,n+1}^0 = \boldsymbol{P}_{1,n}^0$, $\boldsymbol{P}_{m+1,0}^0 = \boldsymbol{P}_{m,1}^0$, $\boldsymbol{P}_{m+1,n+1}^0 = \boldsymbol{P}_{m,n}^0$ and construct the

initial tensor-product surface $\boldsymbol{C}^0(u,v) = \sum_{i=0}^{m+1}\sum_{j=0}^{n+1} \boldsymbol{P}_{i,j}^0 N_{i,3}(u) N_{j,3}(v)$. Assuming that the maximum number of iterations is N, the iteration formula can be written as

$$\boldsymbol{P}_{i,j}^k = \boldsymbol{P}_{i,j}^{k-1} + \boldsymbol{\Delta}_{i,j}^{k-1}, i = 1, \cdots, m; j = 1, \cdots, n; k = 1, \cdots, N, \tag{2}$$

where the difference vector refers to

$$\boldsymbol{\Delta}_{i,j}^{k-1} = (\boldsymbol{Q}_{i,j} - \boldsymbol{C}^{k-1}(u_i, v_j)), i = 1, \cdots, m; j = 1, \cdots, n; k = 1, \cdots, N,$$

and $\boldsymbol{P}_{i,0}^k = \boldsymbol{P}_{i,1}^k$, $\boldsymbol{P}_{i,n+1}^k = \boldsymbol{P}_{i,n}^k$, $\boldsymbol{P}_{0,j}^k = \boldsymbol{P}_{1,j}^k$, $\boldsymbol{P}_{m+1,j}^k = \boldsymbol{P}_{m,j}^k$, $i = 1, \cdots, m, j = 1, \cdots, n$; $\boldsymbol{P}_{0,0}^k = \boldsymbol{P}_{1,1}^k$, $\boldsymbol{P}_{0,n+1}^k = \boldsymbol{P}_{1,n}^k$, $\boldsymbol{P}_{m+1,0}^k = \boldsymbol{P}_{m,1}^k$, $\boldsymbol{P}_{m+1,n+1}^k = \boldsymbol{P}_{m,n}^k$.

For a parameter value, there are only three nonzero basis functions, i.e., there are nine nonzero tensor products at most, and therefore we have

$$\begin{aligned}
\boldsymbol{P}_{i,j}^k =& \boldsymbol{P}_{i,j}^{k-1} + \boldsymbol{\Delta}_{i,j}^{k-1} \\
=& \boldsymbol{P}_{i,j}^{k-1} + (\boldsymbol{Q}_{i,j} - \boldsymbol{C}^{k-1}(u_i, v_j)) \\
=& \boldsymbol{P}_{i,j}^{k-1} + (\boldsymbol{Q}_{i,j} - \boldsymbol{P}_{i-1,j-1}^{k-1} N_{i-1,3}(u_i) N_{j-1,3}(v_j) \\
& - \boldsymbol{P}_{i-1,j}^{k-1} N_{i-1,3}(u_i) N_{j,3}(v_j) - \boldsymbol{P}_{i-1,j+1}^{k-1} N_{i-1,3}(u_i) N_{j+1,3}(v_j) \\
& - \boldsymbol{P}_{i,j-1}^{k-1} N_{i,3}(u_i) N_{j-1,3}(v_j) - \boldsymbol{P}_{i,j}^{k-1} N_{i,3}(u_i) N_{j,3}(v_j) - \boldsymbol{P}_{i,j+1}^{k-1} N_{i,3}(u_i) N_{j+1,3}(v_j) \\
& - \boldsymbol{P}_{i+1,j-1}^{k-1} N_{i+1,3}(u_i) N_{j-1,3}(v_j) - \boldsymbol{P}_{i+1,j}^{k-1} N_{i+1,3}(u_i) N_{j,3}(v_j) - \boldsymbol{P}_{i+1,j+1}^{k-1} N_{i+1,3}(u_i) N_{j+1,3}(v_j)).
\end{aligned}$$

where $i = 1, \cdots, m, j = 1, \cdots, n$.

Similarly, rewrite the iterative format in Equation (2) to matrix form as follows:

$$\boldsymbol{P}^k = (\boldsymbol{I} - \boldsymbol{N})\boldsymbol{P}^{k-1} + \boldsymbol{Q},$$

where $\boldsymbol{I}$ is an identity matrix, $\boldsymbol{N} = \boldsymbol{N}_1 \otimes \boldsymbol{N}_2$ is the Kronecker product between $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$,

$$\boldsymbol{N}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ N_{1,3}(u_2) & N_{2,3}(u_2) & N_{3,3}(u_2) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & N_{m-2,3}(u_{m-1}) & N_{m-1,3}(u_{m-1}) & N_{m,3}(u_{m-1}) \\ 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\boldsymbol{N}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ N_{1,3}(v_2) & N_{2,3}(v_2) & N_{3,3}(v_2) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & N_{n-2,3}(v_{n-1}) & N_{n-1,3}(v_{n-1}) & N_{n,3}(v_{n-1}) \\ 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix},$$

and

$$\begin{aligned}
\boldsymbol{P}^k &= (\boldsymbol{P}_{1,1}^k, \boldsymbol{P}_{1,2}^k, \cdots, \boldsymbol{P}_{1,n}^k, \boldsymbol{P}_{2,1}^k, \boldsymbol{P}_{2,2}^k, \cdots, \boldsymbol{P}_{2,n}^k, \cdots, \boldsymbol{P}_{m,1}^k, \boldsymbol{P}_{m,2}^k, \cdots, \boldsymbol{P}_{m,n}^k), \\
\boldsymbol{Q}^k &= (\boldsymbol{Q}_{1,1}^k, \boldsymbol{Q}_{1,2}^k, \cdots, \boldsymbol{Q}_{1,n}^k, \boldsymbol{Q}_{2,1}^k, \boldsymbol{Q}_{2,2}^k, \cdots, \boldsymbol{Q}_{2,n}^k, \cdots, \boldsymbol{Q}_{m,1}^k, \boldsymbol{Q}_{m,2}^k, \cdots, \boldsymbol{Q}_{m,n}^k).
\end{aligned}$$

2.3. Analysis of Standard PIA Format

Remark 1. *The standard PIA iteration format scheme using nonuniform cubic B-splines can be viewed as decomposing the matrix* $\boldsymbol{N}$ *into* $\boldsymbol{N} = \boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{N})$*, then the following holds*

$$\begin{gathered}
\boldsymbol{NP} = (\boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{N}))\boldsymbol{P} = \boldsymbol{Q} \\
\Downarrow \\
\boldsymbol{P} = (\boldsymbol{I} - \boldsymbol{N})\boldsymbol{P} + \boldsymbol{Q}.
\end{gathered}$$

Replacing $\boldsymbol{P}$ *on the right-hand side with the initial* $\boldsymbol{P}^0$ *will result in a new control point vector* $\boldsymbol{P}^1 = (\boldsymbol{I} - \boldsymbol{N})\boldsymbol{P}^0 + \boldsymbol{Q}$. *Further, substituting the newly generated* $\boldsymbol{P}^1$ *into the right-hand side again will generate another new control vector* $\boldsymbol{P}^2$. *As this process is repeated, we will derive a general iterative form* $\boldsymbol{P}^k = (\boldsymbol{I} - \boldsymbol{N})\boldsymbol{P}^{k-1} + \boldsymbol{Q}, k = 1, 2, \cdots$. *Obviously, the iteration coefficient matrix of the above iterative process should be* $(\boldsymbol{I} - \boldsymbol{N})$.

Lemma 1. *The standard PIA format for non-uniform cubic B-splines converges.*

Proof. Details can be referred to in Ref. [1]. □

Theorem 1. *The standard PIA interpolation for non-uniform cubic B-splines can be regarded as an iterative method of solving linear equations system.*

Proof. Firstly, we know the fact from Remark 1 as follows:

$$\boldsymbol{P}^k = (\boldsymbol{I} - \boldsymbol{N})\boldsymbol{P}^{k-1} + \boldsymbol{Q}, k = 1, 2, \cdots .$$

Then, take the limit of both ends of the above formula and write as

$$\lim_{k\to\infty} \boldsymbol{P}^k = (\boldsymbol{I} - \boldsymbol{N}) \lim_{k\to\infty} \boldsymbol{P}^{k-1} + \boldsymbol{Q}. \tag{3}$$

By Lemma 1, it is determined that the control point sequence definitely converges, and assuming that $\lim_{k\to\infty} \boldsymbol{P}^k = \boldsymbol{P}$, Equation (3) can be finally written as $\boldsymbol{NP} = \boldsymbol{Q}$, which is obviously a system of equations interpolating non-uniform B-spline curve or surface. Thus, the standard PIA iteration format is actually an iterative method for solving a system of linear equations. □

3. The Derivation of SOR-PIA

In this section, we formulate the iterative interpolation process of curves and surfaces on the basis of successive over-relaxation iteration by using non-uniform cubic B-splines as fitting tools. Compared with the standard PIA iteration method [1,2], successive over-relaxation iteration significantly reduces the iteration steps in the iteration process.

3.1. The Case of Curve

The requirements are consistent with the initial conditions of the standard PIA iteration, such as the initial values of three variables, including node vector, knot vector and control point vector. In the standard PIA iterative interpolation process, the calculated $\boldsymbol{P}_1^k, \boldsymbol{P}_2^k, \cdots, \boldsymbol{P}_{i-1}^k$ is utilized to participate in calculating $\boldsymbol{P}_i^k$, then the iterative curve $\boldsymbol{C}^{k-1}(u) = \sum_{p=0}^{m+1} \boldsymbol{P}_p^{k-1} N_{p,3}(u)$ can be replaced by $\boldsymbol{C}^{k,k-1}(u) = \sum_{p=0}^{i-1} \boldsymbol{P}_p^k N_{p,3}(u) + \sum_{p=i}^{m+1} \boldsymbol{P}_p^{k-1} N_{p,3}(u)$, and the iterative scheme of the curve case can be written as

$$\boldsymbol{P}_i^k = \boldsymbol{P}_i^{k-1} + \boldsymbol{\Delta}_i^{k,k-1}, i = 1, \cdots, m; k = 1, 2, \cdots, \tag{4}$$

where the difference vector is denoted as

$$\boldsymbol{\Delta}_i^{k,k-1} = \boldsymbol{Q}_i - \boldsymbol{C}^{k,k-1}(u_i), i = 1, \cdots, m; k = 1, 2, \cdots .$$

We then improve on Equation (4) by adding a specific weight $\frac{\omega}{N_{i,3}(u_i)}$ to the iterative difference vector, where ω is a positive real number, as follows:

$$\boldsymbol{P}_i^k = \boldsymbol{P}_i^{k-1} + \frac{\omega}{N_{i,3}(u_i)} \boldsymbol{\Delta}_i^{k,k-1}, i = 1, \cdots, m; k = 1, 2, \cdots . \tag{5}$$

It differs from traditional PIA by using $\boldsymbol{P}_1^k, \boldsymbol{P}_2^k, \cdots, \boldsymbol{P}_{i-1}^k$ that has been computed before by $\boldsymbol{P}_i^k$ to participate in computing $\boldsymbol{P}_i^k$, and adding a weight $\frac{\omega}{N_{i,3}(u_i)}$ to the iterative difference vector, which includes the relaxation factor ω.

Expand the difference vector in Equation (5) and write it below:

$$\begin{aligned}\boldsymbol{P}_i^k =& \boldsymbol{P}_i^{k-1} + \frac{\omega}{N_{i,3}(u_i)}\boldsymbol{\Delta}_i^{k,k-1}\\ =& \boldsymbol{P}_i^{k-1} + \frac{\omega(\boldsymbol{Q}_i - \boldsymbol{C}^{k,k-1}(u_i))}{N_{i,3}(u_i)}\\ =& (1-\omega)\boldsymbol{P}_i^{k-1} + \omega\frac{\boldsymbol{Q}_i - \boldsymbol{P}_{i-1}^k N_{i-1,3}(u_i) - \boldsymbol{P}_{i+1}^k N_{i+1,3}(u_i)}{N_{i,3}(u_i)}\end{aligned},$$

Thus, Equation (5) can be written in matrix form as follows:

$$\boldsymbol{P}^k = (1-\omega)\boldsymbol{P}^{k-1} + \omega\boldsymbol{D}^{-1}\left(\boldsymbol{Q} - \boldsymbol{L}\boldsymbol{P}^k - \boldsymbol{U}\boldsymbol{P}^{k-1}\right), k = 1, 2, \cdots,$$

where $\boldsymbol{D}$ is a diagonal matrix, $\boldsymbol{L}$ is a strictly lower triangular matrix, $\boldsymbol{U}$ is a strictly upper triangular matrix, and $\boldsymbol{D} + \boldsymbol{L} + \boldsymbol{U} = \boldsymbol{N}$.

3.2. The Case of Surface

Similarly, taking $\boldsymbol{P}_{1,1}^k, \cdots, \boldsymbol{P}_{1,j}^k, \boldsymbol{P}_{2,1}^k, \cdots, \boldsymbol{P}_{2,j}^k, \cdots, \boldsymbol{P}_{i-1,1}^k, \cdots, \boldsymbol{P}_{i-1,j}^k, \boldsymbol{P}_{i,1}^k, \cdots, \boldsymbol{P}_{i,j-1}^k$ that has been obtained before $\boldsymbol{P}_{i,j}^k$ to participate in calculating $\boldsymbol{P}_{i,j}^k$ in the standard PIA iteration process, then we can replace the iterative surface $\boldsymbol{C}^{k-1}(u,v) = \sum_{p=0}^{m+1}\sum_{q=0}^{n+1}\boldsymbol{P}_{p,q}^{k-1}N_{p,3}(u)N_{q,3}(v)$ by $\boldsymbol{C}^{k,k-1}(u,v) = \sum_{p=0}^{i-1}\sum_{q=0}^{n+1}\boldsymbol{P}_{p,q}^{k}N_{p,3}(u)N_{q,3}(v) + \sum_{q=0}^{j-1}\boldsymbol{P}_{i,q}^{k}N_{i,3}(u)N_{q,3}(v) + \sum_{q=j}^{n+1}\boldsymbol{P}_{i,q}^{k-1}N_{i,3}(u)N_{q,3}(v) + \sum_{p=i+1}^{m+1}\sum_{q=0}^{n+1}\boldsymbol{P}_{p,q}^{k-1}N_{p,3}(u)N_{q,3}(v)$. The iterative format of the surface can be summarized as

$$\boldsymbol{P}_{i,j}^k = \boldsymbol{P}_{i,j}^{k-1} + \boldsymbol{\Delta}_{i,j}^{k,k-1}, i = 1, \cdots, m; j = 1, \cdots, n; k = 1, 2, \cdots, \tag{6}$$

where the difference vector refers to

$$\boldsymbol{\Delta}_{i,j}^{k,k-1} = \boldsymbol{Q}_{i,j} - \boldsymbol{C}^{k,k-1}(u_i, v_j), i = 1, \cdots, m; j = 1, \cdots, n; k = 1, 2, \cdots.$$

We then improve on Equation (6) by adding a specific weight $\frac{\omega}{N_{i,3}(u_i)N_{j,3}(v_j)}$ to the iterative difference vector, where ω is a positive real number, as follows:

$$\boldsymbol{P}_{i,j}^k = \boldsymbol{P}_{i,j}^{k-1} + \frac{\omega}{N_{i,3}(u_i)N_{j,3}(v_j)}\boldsymbol{\Delta}_{i,j}^{k,k-1}, i = 1, \cdots, m; j = 1, \cdots, n; k = 1, 2, \cdots, \tag{7}$$

The iterative process expressed in Equations (5) and (7) has great significance in the progressive iterative approximation, which is called SOR-PIA.

Similarly, Equation (7) can also be rewritten as in the case of curves as follows:

$$\begin{aligned}\boldsymbol{P}_{i,j}^k &= \boldsymbol{P}_{i,j}^{k-1} + \frac{\omega}{N_{i,3}(u_i)N_{j,3}(v_j)}\boldsymbol{\Delta}_{i,j}^{k,k-1}\\ &= \boldsymbol{P}_{i,j}^{k-1} + \frac{\omega(\boldsymbol{Q}_i - \boldsymbol{C}^{k,k-1}(u_i, v_j))}{N_{i,3}(u_i)N_{j,3}(v_j)}\\ &= (1-\omega)\boldsymbol{P}_{i,j}^{k-1} + \omega\frac{\left(\begin{array}{c}\boldsymbol{Q}_{i,j} - \boldsymbol{P}_{i,j-1}^k N_{i,3}(u_i)N_{j-1,3}(v_j) - \boldsymbol{P}_{i,j+1}^{k-1}N_{i,3}(u_i)N_{j+1,3}(v_j) -\\ \sum_{q=j-1}^{j+1}((\boldsymbol{P}_{i-1,q}^k N_{i-1,3}(u_i) + \boldsymbol{P}_{i+1,q}^{k-1}N_{i+1,3}(u_i))N_{q,3}(v_j))\end{array}\right)}{N_{i,3}(u_i)N_{j,3}(v_j)}.\end{aligned}$$

When $i = 1, 2, \cdots, m, j = 1, 2, \cdots, n$, Equation (7) can also be written in matrix form

$$\boldsymbol{P}^k = (1-\omega)\boldsymbol{P}^{k-1} + \omega\boldsymbol{D}^{-1}\left(\boldsymbol{Q} - \boldsymbol{L}\boldsymbol{P}^k - \boldsymbol{U}\boldsymbol{P}^{k-1}\right), k = 1, 2, \cdots,$$

where $\boldsymbol{N} = \boldsymbol{N}_1 \otimes \boldsymbol{N}_2$, $\boldsymbol{D}$ is a diagonal matrix, $\boldsymbol{L}$ is a strictly lower triangular matrix, $\boldsymbol{U}$ is a strictly upper triangular matrix, and $\boldsymbol{D} + \boldsymbol{L} + \boldsymbol{U} = \boldsymbol{N}$.

3.3. Computing the Relaxation Factor

According to Sections 3.1 and 3.2, the matrix form of SOR-PIA for non-uniform cubic B-spline interpolation can be uniformly written as

$$\begin{aligned}\boldsymbol{P}^k =& (1-\omega)\boldsymbol{P}^{k-1} + \omega\boldsymbol{D}^{-1}\left(\boldsymbol{Q} - \boldsymbol{L}P^k - \boldsymbol{U}P^{k-1}\right)\\ =& (\boldsymbol{D} + \omega\boldsymbol{L})^{-1}((1-\omega)\boldsymbol{D} - \omega\boldsymbol{U})\boldsymbol{P}^{k-1} + \omega(\boldsymbol{D} + \omega\boldsymbol{L})^{-1}\boldsymbol{Q}, k = 1, 2, \cdots.\end{aligned} \tag{8}$$

Denoting $\boldsymbol{B}_\omega = (\boldsymbol{D} + \omega\boldsymbol{L})^{-1}((1-\omega)\boldsymbol{D} - \omega\boldsymbol{U})$ and $\boldsymbol{l}_\omega = \omega(\boldsymbol{D} + \omega\boldsymbol{L})^{-1}\boldsymbol{Q}$, we can express Equation (8) as

$$\boldsymbol{P}^k = \boldsymbol{B}_\omega\boldsymbol{P}^{k-1} + \boldsymbol{l}_\omega, k = 1, 2, \cdots, \tag{9}$$

where $\boldsymbol{B}_\omega$ is the iteration coefficient matrix and obviously $\rho(\boldsymbol{B}_\omega) \geq |\omega - 1|$. If the sequence of control points $\{\boldsymbol{P}^k\}$ converges, then $|\omega - 1| \leq \rho(\boldsymbol{B}_\omega) < 1$, i.e., $0 < \omega < 2$.

The speed of SOR-PIA is closely related to the relaxation factor. Generally speaking,

(1) $1 < \omega < 2$ means over-relaxation;
(2) $0 < \omega < 1$ means under-relaxation.

At present, there is no explicit method for automatically calculating the exact relaxation factor. This is usually done by taking multiple different values in $[0, 2]$ and comparing them. Our approach provides a way to find an optimal relaxation factor by using an adaptive genetic algorithm. Firstly, we must to determine the evaluation criteria for the effectiveness of each iteration. Let the error of the k-th iterative interpolation of a curve or surface be

$$E_k = \begin{cases} \sum\limits_{i=1}^{m} \|\boldsymbol{C}^{k,k-1}(u_i) - \boldsymbol{Q}_i\|_2 \\ \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} \|\boldsymbol{C}^{k,k-1}(u_i, v_j) - \boldsymbol{Q}_{i,j}\|_2 \end{cases}, k = 1, 2, \cdots. \tag{10}$$

Although the relaxation factor $0 < \omega < 2$ is a necessary condition for iterative convergence, it does not guarantee that all of the relaxation factor in (0, 2) can lead to convergence of the iteration, thus we introduce the average convergence rate to evaluate the effect of the relaxation factor on the convergence rate of the SOR-PIA method. Combined with Equation (10), an objective function for solving the relaxation factor can be constructed as follows, where the variables are the relaxation factor, and the function value is the average convergence rate within arbitrary j iterations starting from the i iteration.

$$g(\omega, i, j) = \frac{1}{j+1}\sum_{k=i}^{i+j}\frac{E_{k+1}}{E_k} \tag{11}$$

We can then use Equation (11) as an objective function to apply the genetic algorithm to search for suitable relaxation factor. In this paper, we take $i = 0$ and $j = 9$ in Equation (11). The reason for our choice is that the rate of convergence in the early stage of convergence is faster than that in the later period and a good solution is easier to stand out. What we need to pay attention to is that the genetic algorithm is only a heuristic algorithm and cannot guarantee the absolute optimal solution, but in the absence of other effective methods, it can be regarded as a feasible scheme.

4. Experiments

In this section, we aim to illustrate the effectiveness of the proposed SORPIA method with several examples and compare it with the PIA method in Refs. [1,2] and the WPIA method in Ref. [6]. These data sets include six synthetic data sets and a real scanned face point cloud data set.

4.1. Iterative Interpolation on Six Synthetic Data

First of all, we provide a detailed description of these synthetic test examples as follows:

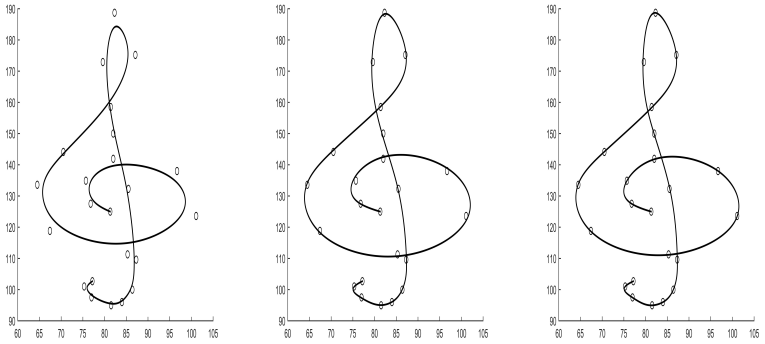
- Curve example 1: 23 data points sampled from the B-spline curve resembling a musical symbol;
- Curve example 2: 12 data points constructed by us, which are in the shape of a two-dimensional spiral;
- Curve example 3: 32 data points sampled on a three-dimensional spiral curve.
- Surface example 1: This is a practical example of a blue and white porcelain. We hope to reconstruct it by interpolating 47 data points.
- Surface example 2: There are 63 sampling data points of the mountain terrain.
- Surface example 3: 121 data points obtained from the transformation of data points sampled from the probability density function of the two-element Gauss distribution.

These data sets are rich enough to represent the majority of the cases, enabling us to have a strong convincing force.

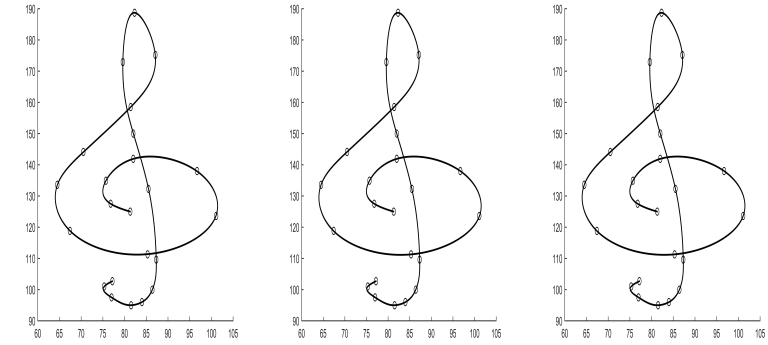
4.1.1. Experimental Results

The final iterative process of PIA, WPIA, and SOR-PIA proposed by us can all be written in the form of $\boldsymbol{P}^k = \boldsymbol{B}\boldsymbol{P}^{k-1} + \boldsymbol{L}, k = 1, 2, \cdots$, which consist of an iterative matrix $\boldsymbol{B}$ and a constant vector $\boldsymbol{L}$ with the same size in the three methods for the same iterative interpolation problem. It can be seen from the above formula, assuming the same experimental conditions, that the running time taken for the computer calculations of each iteration of the three methods is equal. So the iterative speed of the three methods can be estimated by observing the error value of each iteration. Therefore, in the next numerical experiments, the error values of various methods in different iteration level are used as a criterion to judging the speed of convergence.

Before the comparison experiments, we calculated the relaxation factor of SOR-PIA in advance by performing the genetic algorithm solution strategy mentioned in the previous section. In the next subsection, we will discuss the relaxation factor in detail. In order to verify that SOR-PIA is convergent, for each example, we have given the visualizations at different levels of iteration and the final interpolation, as shown in Figures 1–6, and the error value after each iteration is given simultaneously. Since the shape after only a few iterations is very close to that of the complete interpolation, we only show the visualizations of previous iterations.

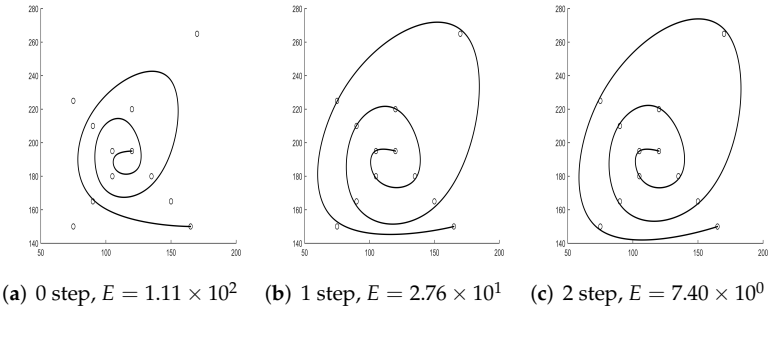


(**a**) 0 step, $E = 3.71 \times 10^{1}$ (**b**) 1 step, $E = 9.88 \times 10^{0}$ (**c**) 2 step, $E = 2.63 \times 10^{0}$

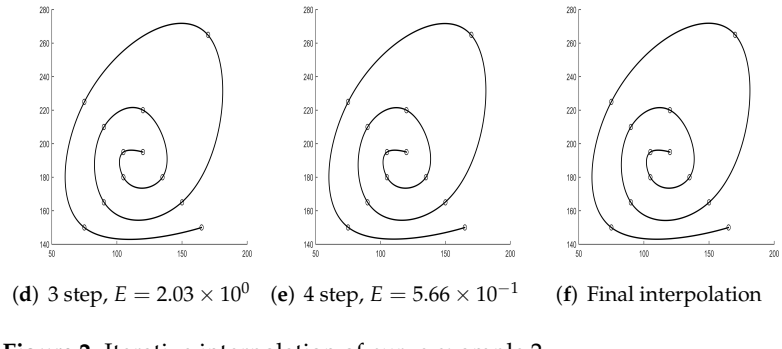


(**d**) 3 step, $E = 2.77 \times 10^{-1}$ (**e**) 4 step, $E = 1.60 \times 10^{-1}$ (**f**) Final interpolation

Figure 1. Iterative interpolation of curve example 1.

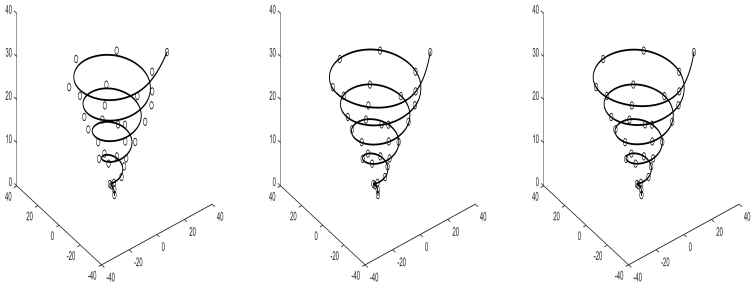


(**a**) 0 step, $E = 1.11 \times 10^{2}$ (**b**) 1 step, $E = 2.76 \times 10^{1}$ (**c**) 2 step, $E = 7.40 \times 10^{0}$

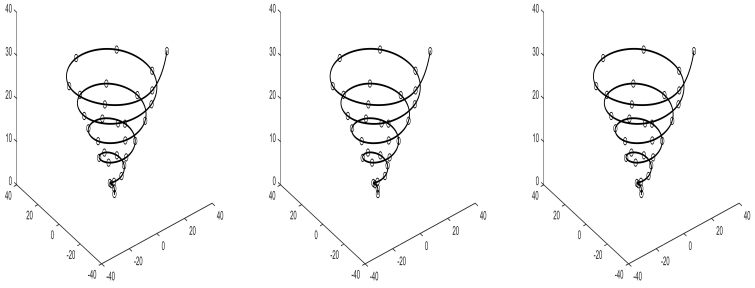


(**d**) 3 step, $E = 2.03 \times 10^{0}$ (**e**) 4 step, $E = 5.66 \times 10^{-1}$ (**f**) Final interpolation

Figure 2. Iterative interpolation of curve example 2.

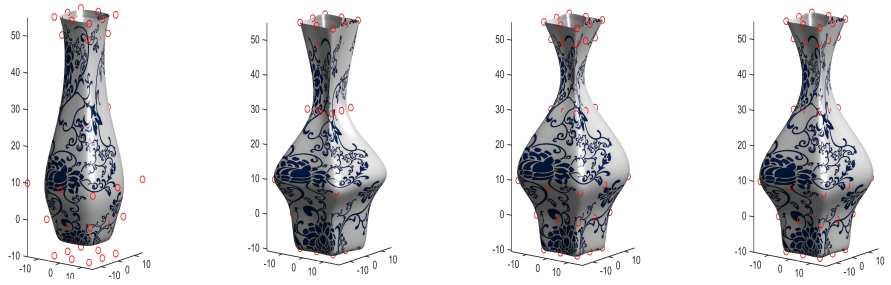


(**a**) 0 step, $E = 1.66 \times 10^2$ (**b**) 1 step, $E = 3.07 \times 10^1$ (**c**) 2 step, $E = 7.69 \times 10^0$

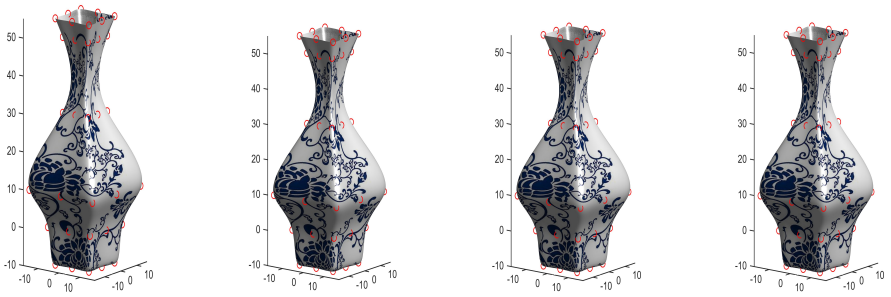


(**d**) 3 step, $E = 1.90 \times 10^0$ (**e**) 4 step, $E = 4.62 \times 10^{-1}$ (**f**) Final interpolation

Figure 3. Iterative interpolation of curve example 3.

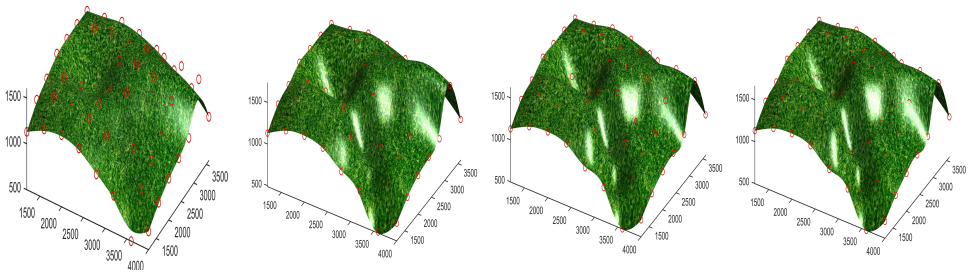


(**a**) 0 step, $E = 1.49 \times 10^2$ (**b**) 1 step, $E = 6.13 \times 10^1$ (**c**) 2 step, $E = 2.79 \times 10^1$ (**d**) 3 step, $E = 1.22 \times 10^1$

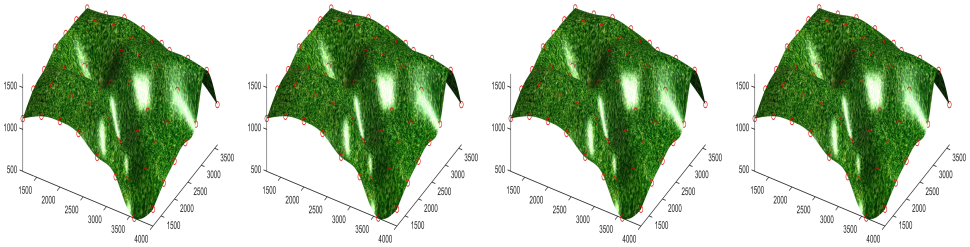


(**e**) 4 step, $E = 5.08 \times 10^0$ (**f**) 5 step, $E = 2.04 \times 10^0$ (**g**) 6 step, $E = 7.48 \times 10^{-1}$ (**h**) Final interpolation

Figure 4. Iterative interpolation of surface example 1.

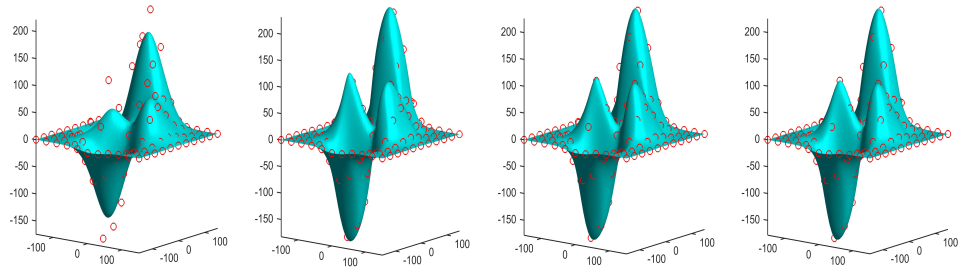


(**a**) 0 step, $E = 3.49 \times 10^3$ (**b**) 1 step, $E = 1.41 \times 10^3$ (**c**) 2 step, $E = 5.74 \times 10^2$ (**d**) 3 step, $E = 2.49 \times 10^2$

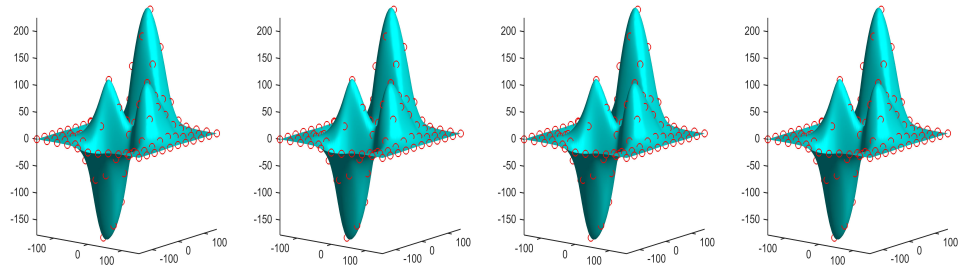


(**e**) 4 step, $E = 1.07 \times 10^2$ (**f**) 5 step, $E = 4.64 \times 10^1$ (**g**) 6 step, $E = 1.69 \times 10^1$ (**h**) Final interpolation

Figure 5. Iterative interpolation of surface example 2.



(**a**) 0 step, $E = 1.09 \times 10^3$ (**b**) 1 step, $E = 5.85 \times 10^2$ (**c**) 2 step, $E = 1.38 \times 10^2$ (**d**) 3 step, $E = 5.08 \times 10^1$

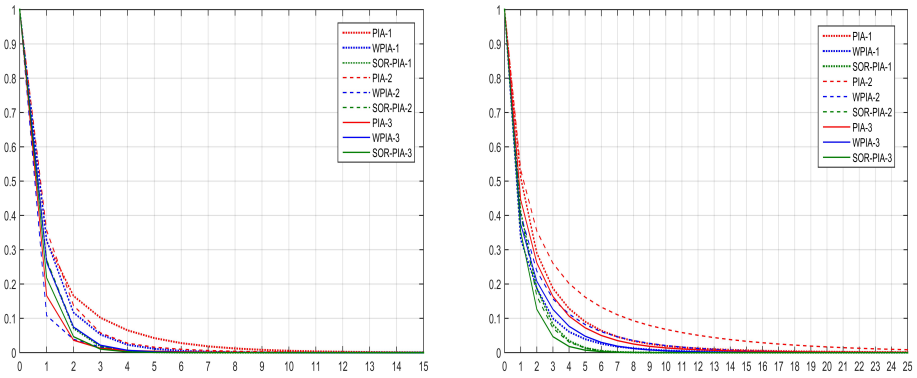


(**e**) 4 step, $E = 1.95 \times 10^1$ (**f**) 5 step, $E = 7.75 \times 10^0$ (**g**) 6 step, $E = 3.19 \times 10^0$ (**h**) Final interpolation

Figure 6. Iterative interpolation of surface example 3.

Figure 7 shows the comparison of errors of various methods in different iteration levels, where PIA is the standard iteration method, WPIA is the PIA method plus an accelerated weight, and SOR-PIA is our method plus a relaxation factor calculated by the genetic algorithm, and three ordinal numbers 1, 2, and 3 correspond to three examples of curves and surfaces, respectively. In view of the different convergence rates of iterative interpolation for curves and surfaces, we choose to observe the convergence of iterative interpolation of curves and surfaces in the 15 and 25 generations, respectively. For an example in Figure 7a, although PIA or WPIA dropped rapidly in the third iteration, SOR-PIA was surpassed after the fourth iteration. In Figure 7b, SOR-PIA is excellent from the beginning to the end. Through the comparison of the convergence process of six examples,

in general, at the same error level, the number of iterations consumed by SOR-PIA is less than that of PIA and WPIA. This shows the feasibility and effectiveness of our method.



(**a**) curve (**b**) surface

Figure 7. Errors comparison among PIA, WPIA and SOR-PIA at different iterative levels.

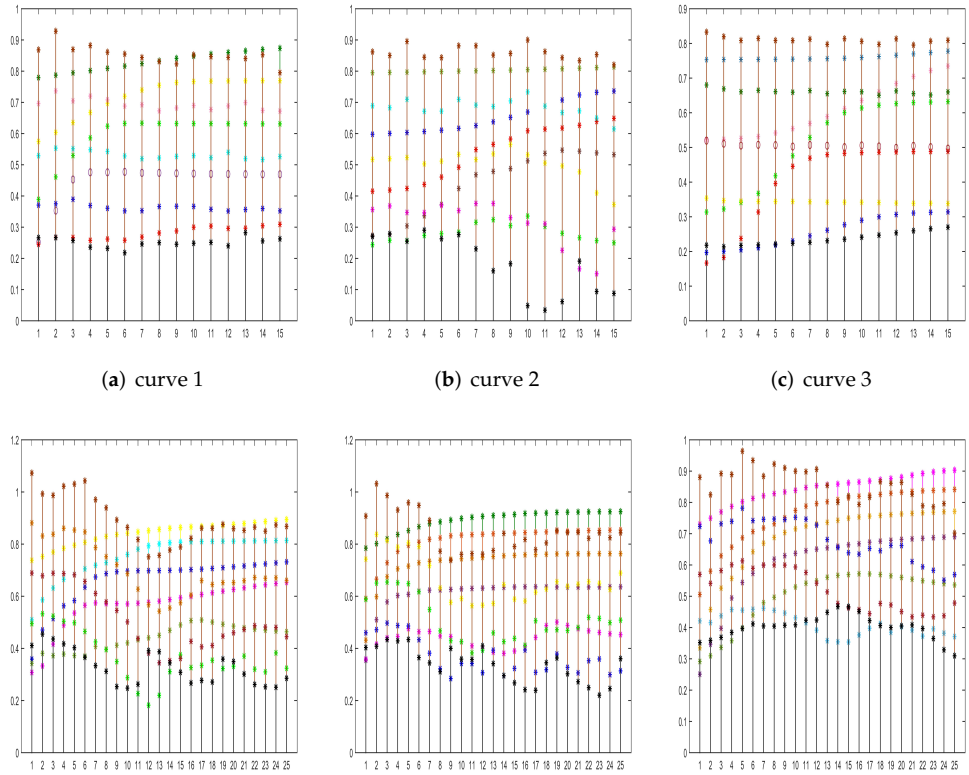
4.1.2. Relaxation Factors

In this paper, the binary coded genetic algorithm is used to calculate the relaxation factor. Owing to the relaxation factor being a single variable and its search range $(0, 2)$ being relatively small, we choose the size of the population and the maximum numbers of genetic generations are 20 and 50, respectively. The program implementations of genetic operators, including selection, crossover, and mutation, can refer to the Matlab GA toolbox, and are not repeated here.

In Table 1, we give the relaxation factors of six examples calculated by the genetic algorithm. For each example, the relaxation factor is calculated only before the iteration, and it is not calculated at every iteration. In addition, to demonstrate that the relaxation factors calculated by the genetic algorithm is approximately optimal, we choose nine values uniformly in the feasible region (0, 2), and use them to test every example. As shown in Figure 7, we choose to observe in 15 and 25 generations, respectively, here. We find that the relaxation factor calculated by the genetic algorithm is the best in the performance of the average convergence rate in Figure 8, which also shows that the genetic algorithm is effective in estimating the relaxation factor. *Note: the convergence rate of each generation is the ratio of the error of the next iteration to the error of the next iteration, which means that the smaller the convergence rate value is, the faster the convergence rate will be. In Figure 8, each color represents a relaxation factor, the ordinate represents the convergence rate, and the abscissa refers to the number of iterations.*

Table 1. Relaxation factors of six examples.

	Example 1	**Example 2**	**Example 3**
curve	1.043 680	1.065 079	1.029 709
surface	1.101 226	1.095 775	1.100 466



(**a**) curve 1 (**b**) curve 2 (**c**) curve 3

(**d**) surface 1 (**e**) surface 2 (**f**) surface 3

Figure 8. The performance of convergence rate with different relaxation factors; black stars represent the case of the relaxation factor calculated by genetic algorithm.

4.1.3. Elapsed Time Comparison

To illustrate the comparison of convergence rates more intuitively, the elapsed time and iteration steps taken by various methods to achieve the same given error precision are recorded here. To make a more comprehensive evaluation, we set different error precision for the six examples, namely, 1×10^{-6}, 1×10^{-9} and 1×10^{-12} for curve and 1×10^{-3}, 1×10^{-6} and 1×10^{-9} for surface. Assuming that the elapsed time and iteration steps are abbreviated as ET and IS, respectively, we present the experimental results in Tables 2 and 3, in which time is measured in seconds. As can be seen from Tables 2 and 3, SOR-PIA requires fewer iterations and less time to achieve the same error precision compared to PIA and WPIA, and has a significant lead.

Table 2. The elapsed time and iteration steps on curve examples.

Curve	Precision	PIA		WPIA		SOR-PIA	
		ET	IS	ET	IS	ET	IS
1	1×10^{-6}	2.42×10^{-3}	37	1.52×10^{-3}	22	6.06×10^{-4}	12
	1×10^{-9}	2.45×10^{-3}	54	2.42×10^{-3}	32	7.51×10^{-4}	16
	1×10^{-12}	3.09×10^{-3}	71	1.89×10^{-3}	41	9.54×10^{-4}	21
2	1×10^{-6}	1.65×10^{-3}	38	9.30×10^{-4}	22	3.56×10^{-4}	10
	1×10^{-9}	1.34×10^{-3}	55	1.72×10^{-3}	32	3.42×10^{-4}	13
	1×10^{-12}	1.69×10^{-3}	71	1.07×10^{-3}	42	3.75×10^{-4}	15
3	1×10^{-6}	1.47×10^{-2}	34	1.55×10^{-2}	20	9.89×10^{-4}	12
	1×10^{-9}	5.40×10^{-3}	50	4.88×10^{-3}	30	2.81×10^{-3}	18
	1×10^{-12}	5.04×10^{-3}	67	3.38×10^{-3}	39	1.46×10^{-3}	23

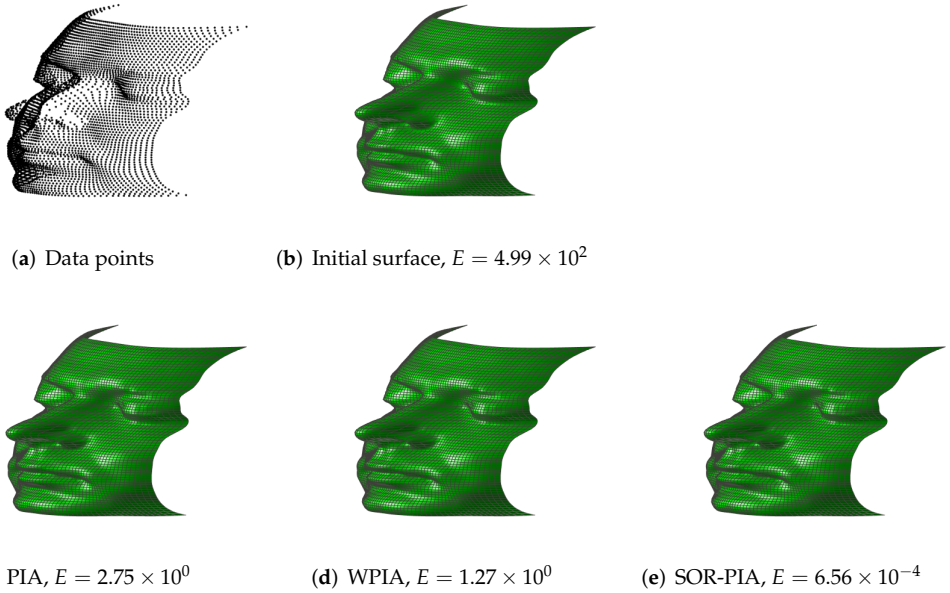
Table 3. The elapsed time and iteration steps on surface examples.

Surface	Precision	PIA		WPIA		SOR-PIA	
		ET	IS	ET	IS	ET	IS
1	1×10^{-3}	3.47×10^{-2}	58	1.97×10^{-2}	31	1.01×10^{-2}	11
	1×10^{-6}	5.89×10^{-2}	107	3.00×10^{-2}	56	9.28×10^{-3}	16
	1×10^{-9}	7.82×10^{-2}	158	4.31×10^{-2}	83	1.11×10^{-2}	22
2	1×10^{-3}	4.36×10^{-2}	100	2.35×10^{-2}	53	6.82×10^{-3}	14
	1×10^{-6}	6.40×10^{-2}	150	3.53×10^{-2}	80	1.04×10^{-2}	20
	1×10^{-9}	8.60×10^{-2}	201	4.60×10^{-2}	107	1.20×10^{-2}	26
3	1×10^{-3}	6.64×10^{-2}	67	9.36×10^{-2}	35	1.53×10^{-2}	14
	1×10^{-6}	1.09×10^{-1}	116	6.13×10^{-2}	61	2.22×10^{-2}	22
	1×10^{-9}	1.58×10^{-1}	166	8.46×10^{-2}	87	2.73×10^{-2}	28

4.2. Iterative Interpolation on the Scanned Face Data

Except for the six synthetic examples given above, we also performed experiments on a scanned face dataset. The original point cloud data contained 84,461 data points, which were simplified and reduced to 4761 data points as shown in Figure 9a for more efficient iteration. More generally, we just take the relaxation factor of SOR-PIA to be 1. Then, the interpolation results of 20-th iteration employing three methods are shown in Figure 9. Unfortunately, due to the density of data points and the free expression of B-splines on surfaces, it is difficult to distinguish the differences between them with the naked eye, so we marked the fitting errors at the corresponding positions. The bar charts of each iteration of three methods within 20 iterations are drawn in Figure 10, from which it can be seen that the convergence rate of SOR-PIA is much higher than that of PIA and WPIA.

Moreover, the elapsed time and iteration steps of the three methods under various error precision in the scanned face dataset are counted in Table 4. Here, the elapsed time not only refers to that of the iterative process, but also includes the elapsed time of calculating the iteration matrix in advance. We can see that WPIA does not necessarily save more time than PIA due to the need to calculate the optimal iteration weight.



(**a**) Data points (**b**) Initial surface, $E = 4.99 \times 10^{2}$

(**c**) PIA, $E = 2.75 \times 10^{0}$ (**d**) WPIA, $E = 1.27 \times 10^{0}$ (**e**) SOR-PIA, $E = 6.56 \times 10^{-4}$

Figure 9. The scanned face data and interpolation results of three methods.

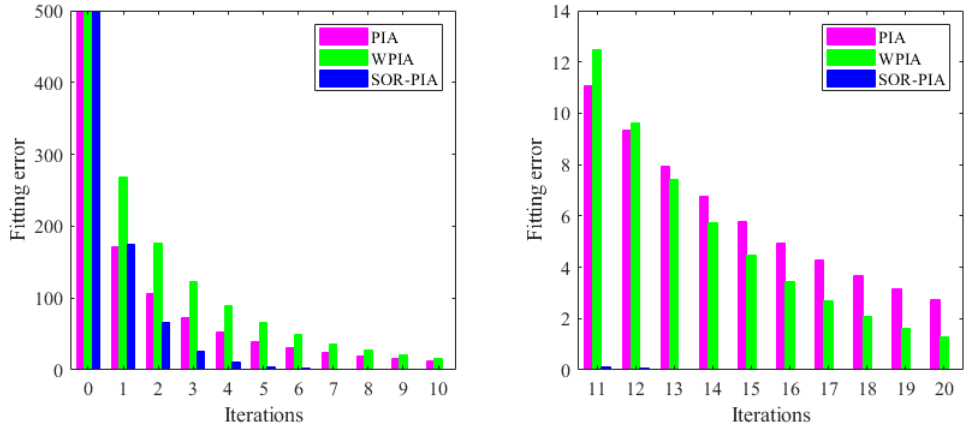


Figure 10. The iterative error comparison of three methods on the scanned face data.

Table 4. The elapsed time and iteration steps on scanned data.

Precision	PIA		WPIA		SOR-PIA	
	ET	IS	ET	IS	ET	IS
1×10^{-3}	4.73×10^{0}	80	2.56×10^{1}	50	3.10×10^{0}	19
1×10^{-6}	8.06×10^{0}	137	2.72×10^{1}	80	3.63×10^{0}	32
1×10^{-9}	1.13×10^{1}	194	2.87×10^{1}	110	4.35×10^{0}	45

5. Conclusions

There are two types of SOR-PIA iterative methods: one is the one-by-one iteration, where the iteration of the second control point needs to follow the iteration of the first control point, the iteration of the third control point need to follow the iteration of the first and second control points, then continues to progress until the iteration of all control points is completed; the other is to calculate the iterative matrix before the iteration, where the iterative process can be written as a matrix form $\boldsymbol{P}^k = \boldsymbol{B}\boldsymbol{P}^{k-1} + \boldsymbol{L}$, and it can perform parallel computing. The disadvantage of the first iterative method is that it cannot perform parallel computing, but only one set of iterative variables is needed in the case such that the PIA must have two sets of iterative variables. Although the second method needs much time to calculate the iterative matrix, the iteration process is very fast owing to the spectral radius of the SOR-PIA iterative matrix being much less than that of PIA.

The above experimental results show that the proposed SOR-PIA method for the iterative interpolation of non-uniform cubic B-spline curves and surfaces is feasible and effective with the same error level, and the number of iterations is much less than that of PIA and WPIA. In the future, we will try to find an effective and quick method to calculate the relaxation factor and prove its convergence theoretically.

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

Funding: This research was funded by the National Natural Science Foundation of China grant number 61572430.

Data Availability Statement: Not applicable.

Acknowledgments: This paper is an extended version of a conference paper [34].

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

Abbreviations

The following abbreviations are used in this manuscript:

GI	Geometric Iteration
PIA	Progressive Iterative Approximation
WPIA	Weighted Progressive Iterative Approximation
SOR	Successive Over-Relaxation
SOR-PIA	Successive Over-Relaxation Progressive Iterative Approximation
NTP	Normalized Totally Positive
HSS	Hermitian and Skew-Hermitian Splitting
EPIA	Extended Progressive Iterative Approximation
LSPIA	Least Squares Progressive Iterative Approximation

References

1. Lin, H.; Wang, G.; Dong, C. Constructing iterative non-uniform B-spline curve and surface to fit data points. *Sci. China Ser. F Inf. Sci.* **2004**, *47*, 315–331. [CrossRef]
2. Lin, H.; Bao, H.; Wang, G. Totally positive bases and progressive iteration approximation. *Comput. Math. with Appl.* **2005**, *50*, 575–586. [CrossRef]
3. Lin, H.; Zhang, Z. An extended iterative format for the progressive-iteration approximation. *Comput. Graph.* **2011**, *35*, 967–975. [CrossRef]
4. Lin, H.; Zhang, Z. An efficient method for fitting large data sets using T-splines. *SIAM J. Sci. Comput.* **2013**, *35*, A3052–A3068. [CrossRef]
5. 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]
6. Carnicer, J.; Delgado, J.; Peña, J. Richardson's iterative method for surface interpolation. *BIT Numer. Math.* **2013**, *53*, 385–396. [CrossRef]
7. Fischer, B.; Reichel, L. A stable Richardson iteration method for complex linear systems. *Numer. Math.* **1989**, *54*, 225–242. [CrossRef]
8. Maekawa, T.; Matsumoto, Y.; Namiki, K. Interpolation by geometric algorithm. *Comput. Aided Des.* **2007**, *39*, 313–323. [CrossRef]
9. Lin, H. The convergence of the geometric interpolation algorithm. *Comput. Aided Des.* **2010**, *42*, 505–508. [CrossRef]
10. Xiong, Y.; Li, G.; Mao, A. Convergence analysis for B-spline geometric interpolation. *Comput. Graph.* **2010**, *36*, 884–891. [CrossRef]
11. Hamza, Y.; Lin, H.; Li, Z. Implicit progressive-iterative approximation for curve and surface reconstruction. *Comput. Aided Geom. Des.* **2020**, *77*, 101817. [CrossRef]
12. Liu, S.; Liu, T.; Hu, L.; Shang, Y.; Liu, X. Variational progressive-iterative approximation for RBF-based surface reconstruction. *Vis. Comput.* **2021**, *37*, 2485–2497. [CrossRef]
13. Yi, Y.; Hu, L.; Liu, C.; Liu, S.; Luo, F. Progressive Iterative Approximation for Extended Cubic Uniform B-Splines with Shape Parameters. *Bull. Malaysian Math. Sci. Soc.* **2021**, *44*, 1813–1836. [CrossRef]
14. Wang, H.; Yuan, L.; Xiong, J.; Mao, J. The relaxed implicit randomized algebraic reconstruction technique for curve and surface reconstruction. *Comput. Graph.* **2022**, *102*, 9–17. [CrossRef]
15. Lu, L. Weighted progressive iteration approximation and convergence analysis. *Comput. Aided Geom. Des.* **2010**, *27*, 129–137. [CrossRef]
16. Deng, C.; Ma, W. Weighted progressive interpolation of Loop subdivision surfaces. *Comput. Aided Des.* **2012**, *44*, 424–431. [CrossRef]
17. Hu, L.; Shou, H.; Fang, S. A PIA optimization algorithm for non-uniform B-spline curves and surfaces. *Int. J. Model. Simul.* **2017** *37*, 1–11. [CrossRef]
18. Gofuku, S.; Tamura, S.; Maekawa, T. Point-tangent/point-normal B-spline curve interpolation by geometric algorithms. *Comput. Aided Des.* **2009**, *41*, 412–422. [CrossRef]
19. Okaniwa, S.; Nasri, A.; Lin, H. Uniform B-spline curve interpolation with prescribed tangent curvature vectors. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 1474–1487. [CrossRef]
20. Kineri, Y.; Endo, S.; Maekawa, T. Surface design based on direct curvature editing. *Comput. Aided Des.* **2014**, *55*, 1–12. [CrossRef]
21. 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**, *67–68*, 107–117. [CrossRef]
22. Martin, T.; Cohen, E.; Kirby, R. M. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. *Comput. Aided Geom. Des.* **2009**, *26*, 648–664. [CrossRef]
23. Lin, H.; Jin, S.; Hu, Q.; Liu, Z. Constructing B-spline solids from tetrahedral meshes for isogeometric analysis. *Comput. Aided Geom. Des.* **2015**, *35-36*, 109–120. [CrossRef]
24. Khobkhun, B.; Prayote, A.; Rakwatin, P.; Dejdumrong, N. Rice Phenology Monitoring Using PIA Time Series MODIS Imagery. In Proceedings of the International Conference Computer Graphics, Imaging and Visualization, Marrakesh, Morocco, 23–25 May 2013; pp. 84–87. [CrossRef]

25. Zhang, Y.; Wang, P.; Bao, F.; Yao, X.; Zhang, C.; Lin, H. A single-image super-resolution method based on progressive-iterative approximation. *IEEE Trans. Multimed.* **2020**, *22*, 1407–1422. [CrossRef]
26. Ebadi, M.; Ebrahimi, A. Video data compression by progressive iterative approximation. *Int. J. Interact. Multimed. Artif. Intell.* **2021**, *6*, 189–195. [CrossRef]
27. Nuntawisuttiwong, T.; Dejdumrong, N. Approximating handwritten curve by using progressive-iterative approximation. In Proceedings of the International Conference Computer Graphics, Imaging and Visualization, Marrakesh, Morocco, 23–25 May 2013; pp. 33–37. [CrossRef]
28. Senawongsa, S.; Dejdumrong, N. An approach to Thai decorative pattern recognition using Bézier curve representation with progressive iterative approximation. In Proceedings of the International Conference Computer Graphics, Imaging and Visualization, Marrakesh, Morocco, 23–25 May 2013; pp. 46–49. [CrossRef]
29. Kuzelka, K.; Marusak, R. Comparison of selected splines for stem form modeling: A case study in Norway spruce. *Ann. For. Res.* **2014**, *57*, 137–148. [CrossRef]
30. Chantakamo, A.; Dejdumrong, N. Conversion of rational Bézier curves into non-rational Bézier curves using progressive iterative approximation. In Proceedings of the International Conference Computer Graphics, Imaging and Visualization, Marrakesh, Morocco, 23–25 May 2013; pp. 38–41. [CrossRef]
31. Lin, H.; Maekawa, T.; Deng, C. Survey on geometric iterative methods and their applications. *Comput. Aided Des.* **2018**, *95*, 40–51. [CrossRef]
32. Hu, L.; Shou, H.; Dai, Z. HSS-iteration-based iterative interpolation of curves and surfaces with NTP bases. In Proceedings of the International Conference on Simulation Tools and Techniques, Chengdu, China, 8–10 July 2019; pp. 374–384. [CrossRef]
33. Hu, L.; Shou, H.; Dai, Z. HSS-iteration-based iterative interpolation of curves and surfaces with NTP bases. *Wirel. Netw.* **2021**, *27*, 3523–3535. [CrossRef]
34. Hu, L.; Shou, H.; Fang, S. Progressive Iterative Approximation of SOR for Non-uniform Cubic B-spline Curve and Surface Interpolation. *Lect. Notes Inst. Comput. Sci. Soc. Inform. Telecommun. Eng.* **2020**, *369*, 339–349. [CrossRef].