The Visual Computer
https://doi.org/10.1007/s00371-023-03090-8

ORIGINAL ARTICLE

# Constrained least square progressive and iterative approximation (CLSPIA) for B-spline curve and surface fitting

Qingjun Chang[1,2] · Weiyin Ma[3] · Chongyang Deng[2]

Accepted: 28 August 2023
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2023

**Abstract**
Combining the Lagrange multiplier method, the Uzawa algorithm, and the *least square progressive and iterative approximation* (LSPIA), we proposed the *constrained least square progressive and iterative approximation* (CLSPIA) to solve the problem of B-spline curve and surface fitting with constraint on data interpolation, i.e., computing the control points of a B-spline curve or surface which interpolates one set of input points while approximating the other set of given points. Compared with the method of solving the linear system directly, CLSPIA has some advantages as it inherits all the nice properties of LSPIA. Because of the data reuse property of LSPIA, CLSPIA reduces a great amount of computation. Using the local property of LSPIA, we can get shape preserving fitting curves by CLSPIA. CLSPIA is efficient for fitting large-scale data sets due to the fact that its computational complexity is linear to the scale of the input data. The many numerical examples in this paper show the efficiency and effectiveness of CLSPIA.

**Keywords** B-spline · Interpolation and approximation · Data fitting · Progressive and iterative approximation (PIA) · Least square progressive and iterative approximation (LSPIA)

## 1 Introduction

Curve and surface fitting plays an extremely important role in industrial applications. It is a fundamental problem in many fields, such as computer-aided design, computer graphics, data visualization, virtual reality, surface modeling, digital image processing, shape modeling, data mining and many related areas [49, 58]. Data fitting consists of data interpolation in which all data should be interpolated, data approximation based on minimizing a certain measure related to all input data, and constrained fitting in which some points of the input data set should be interpolated and others should be approximated [48].

B-splines in their generalization *non-uniform rational B-splines* (NURBS) form a part of the industry standard for the CAD and graphics communities. B-splines are widely used in the CAD/CAM industries for free-form shape representation and data storage due to their simplicity [33]. They are also widely used in data fitting, especially cubic B-splines [43, 45]. In the literature, there exist some empirical methods to determine the parameters of input data points, knots of B-spline and the number of knots. Their optimization leads to complex nonlinear optimization problems; it is, therefore, difficult to solve, but in general they produce good results [43]. When the number of knots, knots of B-spline, and data parameters are determined, the problem of computing the control points of fitting curves or surfaces reduces to solving a system of linear equations [4, 25, 48, 51], which can be solved directly.

Besides solving the linear system directly, the *progressive and iterative approximation* (PIA) [27, 30] and LSPIA [16] are also efficient and intuitive methods in data interpolation and approximation, respectively, especially for large-scale data sets. It avoids solving a linear system directly, and is very flexible because of the local and progressive manner in updating the resulting control points. It thus has attracted increasing attention in recent years [29, 54, 56] and one can find various algorithms for data interpolation and data approximation [16, 21]. The question that naturally arises is,

✉ Chongyang Deng
dcy@hdu.edu.cn

1 Faculty of Informatics, Università della Svizzera italiana, 6900 Lugano, Switzerland

2 School of Sciences, Hangzhou Dianzi University, Hangzhou 310018, China

3 Department of Mechanical Engineering, City University of Hong Kong, Kowloon, Hong Kong, China

Published online: 05 October 2023



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

can we use the idea of PIA and LSPIA to solve the problem of constrained fitting?

Combining the Lagrange multiplier method, with the Uzawa iteration and LSPIA, we propose to find the solution of constrained fitting by CLSPIA. As we will show later, CLSPIA has some advantages over solving the linear system directly. For example, using LSPIA, it is efficient to optimize the parameters of input data points, knots of B-spline and the number of knots; we demonstrate that we can also easily optimize them using CLSPIA.

The structure of this paper is as follows. In Sect. 2, we show some preliminaries and relevant work of the proposed CLSPIA algorithm. In Sect. 3, we describe the CLSPIA algorithm and analyze its convergence. In Sect. 4, we present some experimental results and shape preserving properties of CLSPIA. In Sect. 5, we give the conclusions of this paper.

## 2 Preliminaries and related work

### 2.1 Constrained fitting

For curve fitting with interpolation constraints, Smith et al. [43, 48] proposed a least squares fitting algorithm with weighted constraints in 1974. Given two sets of points **Q** and **R**, one wants to fit a spline curve that approximates the set of points **Q** while interpolating the other set of points **R**. Applying the Lagrange multiplier method, they converted the problem with partial points interpolation into the solution of a linear system:

$$\begin{bmatrix} \mathbf{A}^{\mathsf{T}}\mathbf{W}\mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ \Lambda \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{\mathsf{T}}\mathbf{W}\mathbf{Q} \\ \mathbf{R} \end{bmatrix}, \tag{1}$$

where **Q** is the unconstrained set of points and **A** denotes its corresponding configuration matrix composed of basis functions, **R** is the set of points as interpolation constraints and **B** denotes its corresponding configuration matrix composed of basis functions, **W** is a diagonal matrix which represents the weights, **P** is the set of control points of the fitting curve, and $\Lambda$ represents a supplementary set of variables for solving the constrained system. The solutions of $\Lambda$, **P** are then derived as

$$\Lambda = (\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{W}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}})^{-1}(\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{W}\mathbf{Q} - \mathbf{R}),$$
$$\mathbf{P} = (\mathbf{A}^{\mathsf{T}}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{W}\mathbf{Q} - (\mathbf{A}^{\mathsf{T}}\mathbf{W}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}}\Lambda.$$

The solutions seem to be complex, and thus it is difficult to handle when the size of the set of data points is very large. In 2005, Ke et al. [23] proposed a new constrained fitting method for profile curve reconstruction. They reduced the condition number of the Hessian matrix by using the similarity transformation and, therefore, the numerical stability was significantly improved.

### 2.2 PIA and LSPIA

PIA obtains a series of fitting curves by iteratively adjusting the position of the control points [27, 30, 31, 36, 47]. The concept of the PIA was first proposed for uniform cubic B-spline curve interpolation by Qi et al. [44] in 1975 and was named as "profit and loss correction algorithm," which was proven to be convergent by De Boor [5]. In 2004, Lin et al. [30] found that non-uniform cubic B-splines have the same properties and further proved that curves and surfaces with normalized and totally positive basis also have this property [27]. From then on, PIA is used for various curve and surface interpolation, including uniform cubic B-spline curves [5, 15, 44, 53], non-uniform B-spline curves/surfaces [16, 24, 26, 30, 31, 37, 52], NURBS curves [47], curves/surfaces with normalized and totally positive basis [27], triangular Bernstein-Bézier (B-B) surfaces [7, 57], Doo–Sabin subdivision surfaces [11, 18], loop subdivision surfaces [12, 13, 37, 38], Catmull–Clark subdivision surfaces [10, 37], Wang–Ball curves [6], Said-Ball surfaces on triangular domain [55] and triangular Bézier surfaces [8].

*Least square fitting* (LSF) is a traditional method for B-spline curve fitting [21, 40, 41], and it reduces the problem to a linear system, which can be solved directly. However, when the scale of the linear system is very large, solving the linear system directly is usually not feasible due to numerical instability.

In 2011, Lin and Zhang [31] proposed the *extended PIA* (EPIA) to fit data using *normalized totally positive* (NTP) bases. The limit of the fitting curve/surface is not the same as the one obtained by LSF. In 2014, Deng and Lin [16] proposed a method called LSPIA for data fitting of B-spline curve and surface by combining the LSF and PIA. Similar to the typical PIA algorithm, LSPIA starts with an initial B-spline curve or surface and constructs a series of fitting curves or surfaces by adjusting the control points iteratively. In each iteration, the adjusting vector of each control point is a weighted sum of some difference vectors between the data points and their corresponding points on the fitting curve or surface) [16]. The limit curve (surface) of LSPIA is the same as the LSF result of the given data points. Zhang et al. [54] showed that the LSPIA is also convergent for generalized B-spline curves with two different kinds of weights. Lin et al. [28] showed that even if the iterative matrix is singular, the LSPIA still converges. Since the iteration of LSPIA is related to the number of data points and is independent of the number of control points, which is unknown, it is suitable for large-scale data processing [29].



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

### 2.3 Uzawa algorithm

Actually, solving (1) is also known as solving a saddle point problem. A saddle point or minimax point [19] is a point on the graph of a function where the slopes (derivatives) in orthogonal directions are both zero (a critical point), but it is not a local extremum of the function [22]. See the survey paper by Benzi et al. [3], there are many classic methods for numerical solution of saddle point problems. We select the Uzawa algorithm [50] as it is a simple and highly efficient method [2, 14, 17, 32, 34, 35, 39] for solving the saddle point problem based on matrix splitting to solve the quadratic optimization problems, to solve (1). For a saddle point problem of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \tag{2}$$

with initial values $\mathbf{x}_1^{(0)}$, $\mathbf{x}_2^{(0)}$, where $\mathbf{A}$ is an invertible matrix, Uzawa algorithm is defined as follows [9, 50]:

$$\mathbf{A}\mathbf{x}_1^{(\mathrm{r})} = \mathbf{b}_1 - \mathbf{B}^{\mathsf{T}}\mathbf{x}_2^{(\mathrm{r}-1)}, \tag{3}$$

$$\mathbf{x}_2^{(\mathrm{r})} = \mathbf{x}_2^{(\mathrm{r}-1)} + \alpha(\mathbf{B}\mathbf{x}_1^{(\mathrm{r})} - \mathbf{b}_2), \tag{4}$$

where $\alpha$ is a given real number, and according to (2), we can get $\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{x}_2 = \mathbf{B}\mathbf{A}^{-1}\mathbf{b}_1 - \mathbf{b}_2$. Uzawa algorithm is equivalent to applying a gradient algorithm to this equation using a fixed step size $\alpha$, and the iteration converges if $\alpha < 2\|\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathsf{T}}\|^{-1}$ [1, 20], where $\|\cdot\|$ denotes a certain norm of the matrix (i.e., the $\ell_2$-norm). See Algorithm 1.

**Algorithm 1** Uzawa Algorithm

**Require:** Given $\alpha < 2\|\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathsf{T}}\|^{-1}$, initialize $\mathbf{x}_2$
**Ensure:** Find the numerical solution $\mathbf{x}_1$, $\mathbf{x}_2$ of (2).
  **repeat**
    $\mathbf{x}_1 \leftarrow \mathbf{A}^{-1}(\mathbf{b}_1 - \mathbf{B}^{\mathsf{T}}\mathbf{x}_2)$
    $\mathbf{x}_2 \leftarrow \mathbf{x}_2 + \alpha(\mathbf{B}\mathbf{x}_1 - \mathbf{b}_2)$
  **until** satisfy convergence criterion

## 3 CLSPIA for curve and surface fitting

From (1), (2), (3) and (4), we can see that the Uzawa algorithm can be used to solve constrained data fitting of B-spline curves and surfaces. Specifically, we use LSPIA to solve the linear system (3) and then the constrained fitting problem (1), we name such algorithm as CLSPIA.

In this section, we first use the Lagrange multiplier method to convert the constrained fitting problem to a saddle point problem in Sect. 3.1, then present the CLSPIA algorithm of B-spline curve fitting in Sect. 3.2, and analyze its convergence in Sect. 3.3. We will extend the CLSPIA method to surface fitting in Sect. 3.4.

### 3.1 Constrained fitting and Lagrange multiplier method

Let $\{K_i\}_{i=0}^{m}$ be an ordered set of points in $\mathbb{R}^2$, $\{Q_j\}_{j=0}^{m_1} \subset \{K_i\}_{i=0}^{m}$ with parameters $\{t_j\}_{j=0}^{m_1}$ being the set of points to be approximated, and let $\{R_k\}_{k=0}^{m_2} \subset \{K_i\}_{i=0}^{m}$ with parameters $\{s_k\}_{k=0}^{m_2}$ be the set of points to be interpolated, where $m_1 + m_2 = m + 1$ and the set $\{Q_j\}_{j=0}^{m_1}$ and set $\{R_k\}_{k=0}^{m_2}$ constitute the complete set $\{K_i\}_{i=0}^{m}$. Let $\mathcal{P}(t)$ be the fitting curve given by

$$\mathcal{P}(t) = \sum_{i=0}^{n} \mathcal{B}_i(t) P_i\,, \quad t \in [0, 1]\,,$$

where $\mathcal{B}_i(t)$ are B-spline basis functions, and $\{P_i\}_{i=0}^{n}$ is the set of control points. The constrained fitting problem is defined as

$$\mathbf{P} = \arg\min_{\mathbf{P}} \sum_{j=0}^{m_1} \left\|\mathcal{P}(t_j) - Q_j\right\|_2^2,$$
$$s.t.\ \mathcal{P}(s_k) = R_k,\ (k = 0, 1, \ldots, m_2),$$

where $\mathbf{P} = [P_0, P_1, \ldots, P_n]^{\mathsf{T}}$. Using the Lagrange multiplier method, the solution can be derived by finding the minimum of

$$\mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) = \sum_{j=0}^{m_1} \left\|\mathcal{P}(t_j) - Q_j\right\|_2^2 + \sum_{k=0}^{m_2} \lambda_k^{\mathsf{T}} \left(\mathcal{P}(s_k) - R_k\right),$$

where $\lambda_k \in \mathbb{R}^2$ and $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_{m_2}]^{\mathsf{T}}$. According to the principle of Lagrange multiplier method, the following linear system can be obtained

$$\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{P} - \mathbf{A}^{\mathsf{T}}\mathbf{Q} + \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda} = \mathbf{0},$$
$$\mathbf{B}\mathbf{P} - \mathbf{R} = 0, \tag{5}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathcal{B}_0(t_0) & \mathcal{B}_1(t_0) & \cdots & \mathcal{B}_n(t_0) \\ \mathcal{B}_0(t_1) & \mathcal{B}_1(t_1) & \cdots & \mathcal{B}_n(t_1) \\ \vdots & \vdots & & \vdots \\ \mathcal{B}_0(t_{m_1}) & \mathcal{B}_1(t_{m_1}) & \cdots & \mathcal{B}_n(t_{m_1}) \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} \mathcal{B}_0(s_0) & \mathcal{B}_1(s_0) & \cdots & \mathcal{B}_n(s_0) \\ \mathcal{B}_0(s_1) & \mathcal{B}_1(s_1) & \cdots & \mathcal{B}_n(s_1) \\ \vdots & \vdots & & \vdots \\ \mathcal{B}_0(s_{m_2}) & \mathcal{B}_1(s_{m_2}) & \cdots & \mathcal{B}_n(s_{m_2}) \end{bmatrix},$$



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

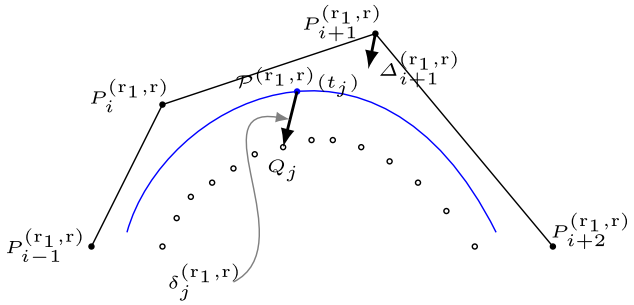



**Fig. 1** Difference vectors $\delta_j$ for all the approximation points (hollow dots) and the adjusting vectors $\Delta_i$ for all control points after the r-th curve $\mathcal{P}^{(r_1,r)}(t)$ (blue) in the $r_1$-th Uzawa iteration

$$\mathbf{Q} = [Q_0, Q_1, \ldots, Q_{m_1}]^{\mathsf{T}}, \ \mathbf{R} = [R_0, R_1, \ldots, R_{m_2}]^{\mathsf{T}}.$$

This leads to a saddle problem

$$\begin{bmatrix} \mathbf{A}^{\mathsf{T}}\mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{\mathsf{T}}\mathbf{Q} \\ \mathbf{R} \end{bmatrix}, \tag{6}$$

based on Uzawa algorithm [50], and the above saddle problem can be solved by the following iterative method:

$$\begin{aligned} &\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{P}^{(r_1)} = \mathbf{A}^{\mathsf{T}}\mathbf{Q} - \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}^{(r_1-1)}, \\ &\boldsymbol{\lambda}^{(r_1)} = \boldsymbol{\lambda}^{(r_1-1)} + \mu(\mathbf{B}\mathbf{P}^{(r_1)} - \mathbf{R}). \end{aligned} \tag{7}$$

### 3.2 The CLSPIA algorithm for curve fitting

At the beginning of the iteration, we specify a set of initial control points $\{P_i^{(1,0)}\}_{i=0}^n$, where the first and second superscript $(1, 0)$ of $P_i$ represent the iteration steps of Uzawa algorithm and LSPIA, respectively, and the initial vector $\lambda_k^{(0)} = [1, 1]^{\mathsf{T}}$ for each interpolation point $R_k$ as the initial value. Then, the initial curve is given as

$$\mathcal{P}^{(1,0)}(t) = \sum_{i=0}^{n} \mathcal{B}_i(t) P_i^{(1,0)}, \quad t \in [0, 1].$$

Our CLSPIA algorithm includes two alternating steps. We first introduce how to use LSPIA to solve (7). Suppose that we have obtained the r-th curve $\mathcal{P}^{(r_1,r)}(t)$ in the $r_1$-th Uzawa iteration, the difference vectors are

$$\delta_j^{(r_1,r)} = Q_j - \mathcal{P}^{(r_1,r)}(t_j), \quad j = 0, 1, \ldots, m_1,$$

then the adjusting vector for the $i$-th control point is defined as (see Fig. 1)

$$\Delta_i^{(r_1,r)} = \frac{1}{\mu_1}(\epsilon_i^{(r_1,r)} - \theta_i^{(r_1)}), \tag{8}$$

where $\mu_1$ is a constant satisfying the condition $\mu_1 > \frac{\gamma_0}{2}$, let $\gamma_0$ be the largest eigenvalue of $\mathbf{A}^{\mathsf{T}}\mathbf{A}$, and

$$\epsilon_i^{(r_1,r)} = \sum_{j=0}^{m_1} \mathcal{B}_i(t_j)\delta_j^{(r_1,r)},$$

$$\theta_i^{(r_1)} = \sum_{k=0}^{m_2} \mathcal{B}_i(s_k)\lambda_k^{(r_1-1)}.$$

The control points are then updated as

$$P_i^{(r_1,r+1)} = P_i^{(r_1,r)} + \Delta_i^{(r_1,r)}, \tag{9}$$

and the $(r + 1)$-th curve can be generated as

$$\mathcal{P}^{(r_1,r+1)}(t) = \sum_{i=0}^{n} \mathcal{B}_i(t) P_i^{(r_1,r+1)}.$$

By induction on r, we get a curve sequence $\{\mathcal{P}^{(r_1,r)}\}_{r=0}^{\infty}$ (see Algorithm 2), and we will prove its convergence in Sect. 3.3. In this way, we can obtain $P_i^{(r_1)} = P_i^{(r_1,\infty)}$ by iteration.

Once we have $P_i^{(r_1)}$ for each $i$, let $P_i^{(r_1+1,0)}$ be $P_i^{(r_1)}$, the corresponding curve is $\mathcal{P}^{(r_1+1,0)}(t)$, then

$$\lambda_j^{(r_1)} = \lambda_j^{(r_1-1)} + \mu(\mathcal{P}^{(r_1+1,0)}(s_j) - R_j), \tag{10}$$

where $\mu$ satisfies the condition $\frac{1}{\mu} > \frac{\beta_0}{2}$, in which, $\beta_0$ is the largest eigenvalue of $\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}}$. Up to now, we have updated the control points $\mathbf{P}^{(r_1+1,0)}$ from $\mathbf{P}^{(r_1,0)}$ and parameters $\boldsymbol{\lambda}^{(r_1)}$ from $\boldsymbol{\lambda}^{(r_1-1)}$, thus one iteration of Uzawa algorithm is completed. By induction on $r_1$, we can get the Uzawa iterative solution $\mathbf{P}$ of the saddle point problem in (6). A general form of CLSPIA algorithm can be written as

$$\begin{aligned} &\mathbf{P}^{(r_1,r+1)} = \mathbf{P}^{(r_1,r)} + \boldsymbol{\Delta}^{(r_1,r)}, \\ &\boldsymbol{\Delta}^{(r_1,r)} = \frac{1}{\mu_1}(\mathbf{A}^{\mathsf{T}}\mathbf{Q} - \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}^{(r_1-1)} - \mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{P}^{(r_1,r)}), \\ &\boldsymbol{\lambda}^{(r_1)} = \boldsymbol{\lambda}^{(r_1-1)} + \mu(\mathbf{B}\mathbf{P}^{(r_1)} - \mathbf{R}), \end{aligned}$$

where $\mathbf{P}^{(r_1)} = \mathbf{P}^{(r_1,\infty)}$. Next, we will show the iterative method is convergent.

### 3.3 Convergence analysis

In this subsection, we discuss the convergence of the iterative algorithm.

**Theorem 1** *The iterative method defined by*

$$\mathbf{P}^{(r_1,r+1)} = \mathbf{P}^{(r_1,r)} + \boldsymbol{\Delta}^{(r_1,r)}$$

*is convergent.*



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

**Algorithm 2** CLSPIA Algorithm

**Require:** Given $\mu$ and $\mu_1$, initialize $\boldsymbol{\lambda}$ and $\mathbf{P}^{(1,0)}$
**Ensure:** Find the numerical solution $\mathbf{P}$ of (6)
**for** $\mathrm{r}_1 = 1, 2, \ldots$ **do**
  **for** $\mathrm{r} = 0, 1, \ldots$ **do**
    $\mathcal{P}^{(\mathrm{r}_1,\mathrm{r})}(t) \leftarrow \sum_{i=0}^{n} \mathcal{B}_{i,l}(t) P_i^{(\mathrm{r}_1,\mathrm{r})}$
    For all $j$, compute $\delta_j^{(\mathrm{r}_1,\mathrm{r})} \leftarrow Q_j - \mathcal{P}^{(\mathrm{r}_1,\mathrm{r})}(t_j)$
    For all $i$, compute $\Delta_i^{(\mathrm{r}_1,\mathrm{r})}$ by (8)
    For all $i$, update $P_i^{(\mathrm{r}_1,\mathrm{r}+1)} \leftarrow P_i^{(\mathrm{r}_1,\mathrm{r})} + \Delta_i^{(\mathrm{r}_1,\mathrm{r})}$
    **if** all $\Delta_i^{(\mathrm{r}_1,\mathrm{r})}$ satisfy convergence criterion **then**
      **return** all $P_i^{(\mathrm{r}_1,\mathrm{r}+1)}$
    **end if**
  **end for**
  For all $i$, initialize $P_i^{(\mathrm{r}_1+1,0)} \leftarrow P_i^{(\mathrm{r}_1,\mathrm{r})}$
  For all $j$, update $\lambda_j$ by (10)
**end for**

***Proof*** Let $\mathbf{D} = \mathbf{I} - \frac{1}{\mu_1}\mathbf{A}^{\mathsf{T}}\mathbf{A}$, where $\mathbf{I}$ is an identity matrix, we have

$$\begin{aligned}
&\mathbf{P}^{(\mathrm{r}_1,\mathrm{r}+1)} + (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}(\mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}^{(\mathrm{r}_1-1)} - \mathbf{A}^{\mathsf{T}}\mathbf{Q}) \\
&\quad = \mathbf{D}[\mathbf{P}^{(\mathrm{r}_1,\mathrm{r})} + (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}(\mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}^{(\mathrm{r}_1-1)} - \mathbf{A}^{\mathsf{T}}\mathbf{Q})] \\
&\quad = \mathbf{D}^{\mathrm{r}+1}[\mathbf{P}^{(\mathrm{r}_1,0)} + (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}(\mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}^{(\mathrm{r}_1-1)} - \mathbf{A}^{\mathsf{T}}\mathbf{Q})].
\end{aligned}$$

Let $\{\gamma_i(\mathbf{D})\}(i = 0, 1, \ldots, n)$ denote the eigenvalues of $\mathbf{D}$, then $\gamma_i(\mathbf{D}) = 1 - \gamma_i/\mu_1$, where $\gamma_i$ is one of the eigenvalues of $\mathbf{A}^{\mathsf{T}}\mathbf{A}$. Since $m_1 > n$, matrix $\mathbf{A}$ is a column full rank matrix, $\mathbf{A}^{\mathsf{T}}\mathbf{A}$ is a symmetric positive definite matrix, and $\mu_1 > \frac{\gamma_0}{2}$, we have

$$1 - \gamma_i/\mu_1 \in (-1, 1),$$

that is, $\rho(\mathbf{D}) < 1$, where $\rho(\mathbf{D})$ is spectral radius of $\mathbf{D}$. This means that the above iterative method converges and

$$\mathbf{P}^{(\mathrm{r}_1,\infty)} = (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}(\mathbf{A}^{\mathsf{T}}\mathbf{Q} - \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}^{(\mathrm{r}_1-1)}). \tag{11}$$

It happens to be the theoretical solution of (7). □

**Theorem 2** *The iterative method defined by*

$$\boldsymbol{\lambda}^{(r_1)} = \boldsymbol{\lambda}^{(r_1-1)} + \mu(\mathbf{B}\mathbf{P}^{(r_1,\infty)} - \mathbf{R}) \tag{12}$$

*is convergent, where $\mathbf{P}^{(r_1,\infty)}$ is the result of the previous iteration. Also, see Corollary 8.1 by Saad* [46].

***Proof*** Combining (11) and (12), let

$$\begin{aligned}
\mathbf{M} &= \left[\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}}\right]^{-1}\left[\mathbf{R} - \mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{Q}\right], \\
\mathbf{D} &= \mathbf{I} - \mu\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}},
\end{aligned}$$

we have

$$\boldsymbol{\lambda}^{(\mathrm{r}_1+1)} + \mathbf{M} = \mathbf{D}(\boldsymbol{\lambda}^{(\mathrm{r}_1)} + \mathbf{M}) = \mathbf{D}^{\mathrm{r}_1+1}(\boldsymbol{\lambda}^{(0)} + \mathbf{M}).$$

Let $\{\beta_i(\mathbf{D})\}(i = 0, 1, \ldots)$ denote one of the eigenvalues of $\mathbf{D}$, we have $\beta_i(\mathbf{D}) = 1 - \mu\beta_i$, where $\beta_i$ is the corresponding eigenvalue of $\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}}$. Since the matrix $\mathbf{A}^{\mathsf{T}}\mathbf{A}$ is a symmetric positive definite matrix, and thus $(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}$ is also a symmetric positive definite matrix. Thus, there exists a real reversible matrix $\mathbf{C}$, such that $(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1} = \mathbf{C}\mathbf{C}^{\mathsf{T}}$. Then, we have $\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}} = (\mathbf{B}\mathbf{C})(\mathbf{B}\mathbf{C})^{\mathsf{T}}$, and $\mathbf{B}\mathbf{C}$ is a non-singular matrix. So $(\mathbf{B}\mathbf{C})(\mathbf{B}\mathbf{C})^{\mathsf{T}}$ is a positive definite matrix, and thus $\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}}$ is also a positive definite matrix. At the same time, since $\frac{1}{\mu} > \frac{\beta_0}{2}$ and $\beta_0$ is the largest eigenvalue of $\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}}$, for each $i$, we have $\mu\beta_i \in (0, 2)$. Therefore, $\beta_i(\mathbf{D}) \in (-1, 1)$, and $\rho(\mathbf{D}) < 1$. This means that $\boldsymbol{\lambda}^{(\infty)} = -\mathbf{M}$,

$$\boldsymbol{\lambda}^{(\infty)} = \left[\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}}\right]^{-1}[\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{Q} - \mathbf{R}],$$

thus the above iteration converges. □

### 3.4 CLSPIA for surface fitting

The CLSPIA introduced in Sects. 3.2 and 3.3 can be extended to the case of tensor product surface fitting. For the remainder of this section, we provide further details for this case.

Assume that $\{K_i\}_{i=0}^{m}$ is an ordered set of points which need to be fitted, $\{Q_i\}_{i=0}^{m_1} \subset \{K_i\}_{i=0}^{m}$ with parameters $\{(t_{u_i}, t_{v_i})\}$ is the set of points to be approximated, and $\{R_i\}_{i=0}^{m_2} \subset \{K_i\}_{i=0}^{m}$ with parameters $\{(s_{u_i}, s_{v_i})\}$ is the set of points to be interpolated. To define the fitting surface $\mathcal{P}(u, v)$ in domain $(u, v) \in [0, 1]^2$ as

$$\mathcal{P}(u, v) = \sum_{h=0}^{n_1}\sum_{l=0}^{n_2} \mathcal{B}_h(u)\mathcal{B}_l(v)P_{h,l},$$

where $P_{h,l}$ is a set of control points, $\mathcal{B}_h(u)$ and $\mathcal{B}_l(v)$ are the basis functions in two dimensions of the tensor product, respectively. Then, the constrained fitting problem is defined as

$$\begin{aligned}
\mathbf{P} = \arg\min_{\mathbf{P}} &\sum_{i=0}^{m_1} \left\|\mathcal{P}(t_{u_i}, t_{v_i}) - Q_i\right\|^2 \\
&s.t.\ \mathcal{P}(s_{u_i}, s_{v_i}) = R_i, \quad (i = 0, 1, \ldots, m_2).
\end{aligned}$$

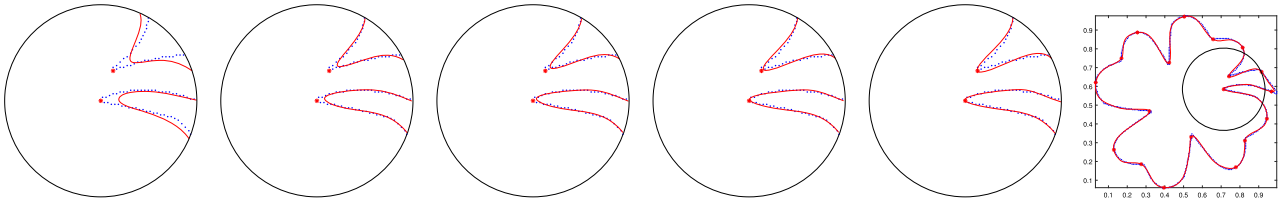
where $\mathbf{P} = [P_{0,0}, P_{0,1}, \ldots, P_{n_1,n_2}]^{\mathsf{T}}$. Let us define matrices $\mathbf{A} \in \mathbb{R}^{(m_1+1)\times(n_1n_2+1)}$ and $\mathbf{B} \in \mathbb{R}^{(m_2+1)\times(n_1n_2+1)}$ with

$$\begin{aligned}
A_{i,j} &= \mathcal{B}_{j\oslash(n_2+1)}(t_{u_i})\mathcal{B}_{j\odot(n_2+1)}(t_{v_i}), \\
B_{i,j} &= \mathcal{B}_{j\oslash(n_2+1)}(s_{u_i})\mathcal{B}_{j\odot(n_2+1)}(s_{v_i}),
\end{aligned}$$

where $\oslash$ and $\odot$ denote quotient and remainder, respectively. Similar to the curve fitting case mentioned before, by

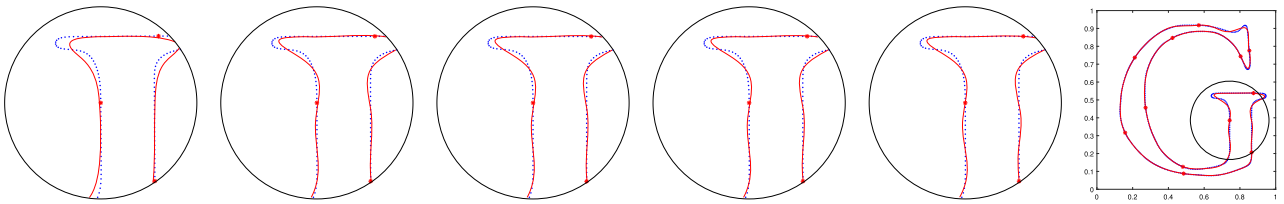


Content courtesy of Springer Nature, terms of use apply. Rights reserved.



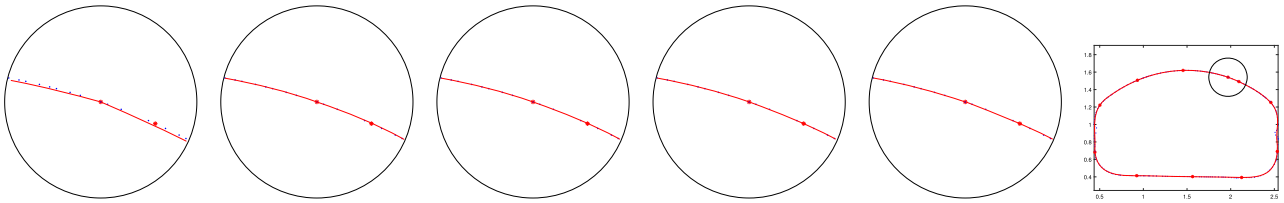
(a) Zoom in of step 0 (b) Zoom in of step 1 (c) Zoom in of step 3 (d) Zoom in of step 7 (e) Zoom in of the limit (f) Final curve

**Fig. 2** A cubic B-spline curve with 50 control points fitting 501 input data points while interpolating 19 given points.



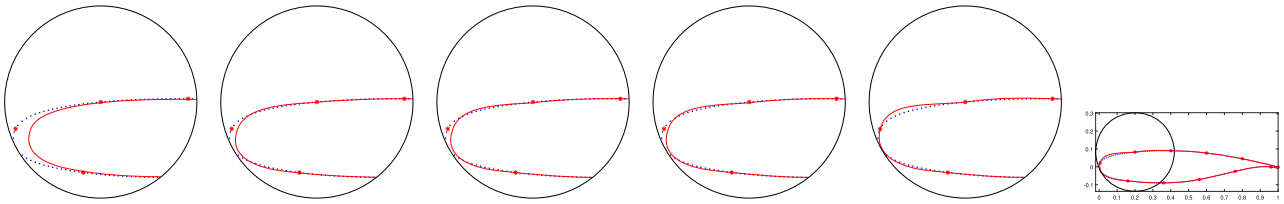
(a) Zoom in of step 0 (b) Zoom in of step 1 (c) Zoom in of step 3 (d) Zoom in of step 7 (e) Zoom in of the limit (f) Final curve

**Fig. 3** A cubic B-spline curve with 55 control points fitting 577 input data points while interpolating 12 given points



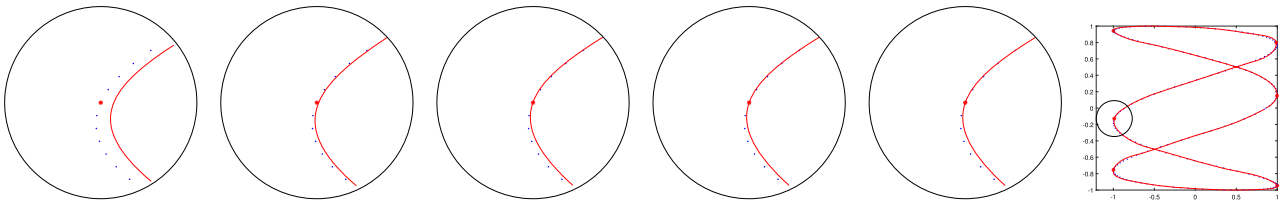
(a) Zoom in of step 0 (b) Zoom in of step 1 (c) Zoom in of step 3 (d) Zoom in of step 7 (e) Zoom in of the limit (f) Final curve

**Fig. 4** A cubic B-spline curve with 30 control points fitting 205 input data points while interpolating 11 given points



(a) Zoom in of step 0 (b) Zoom in of step 1 (c) Zoom in of step 3 (d) Zoom in of step 7 (e) Zoom in of the limit (f) Final curve

**Fig. 5** An illustration of local shape fitting under different iteration steps for fitting a B-spline curve to 205 data points using 20 control points with interpolation constraints

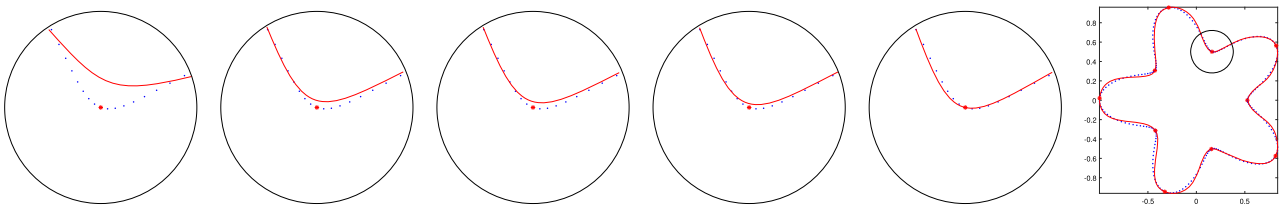


(a) Zoom in of step 0 (b) Zoom in of step 1 (c) Zoom in of step 3 (d) Zoom in of step 7 (e) Zoom in of the limit (f) Final curve

**Fig. 6** An illustration of local shape fitting under different iteration steps for fitting a B-spline curve to 210 data points using 37 control points with interpolation constraints



Content courtesy of Springer Nature, terms of use apply. Rights reserved.



(a) Zoom in of step 0 (b) Zoom in of step 1 (c) Zoom in of step 3 (d) Zoom in of step 7 (e) Zoom in of the limit (f) Final curve

**Fig. 7** An illustration of local shape fitting under different iteration steps for fitting a B-spline curve to 210 data points using 22 control points with interpolation constraints

applying the Lagrange multiplier method, we get

$$\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{P} - \mathbf{A}^{\mathsf{T}}\mathbf{Q} + \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda} = \mathbf{0},$$
$$\mathbf{B}\mathbf{P} = \mathbf{R},$$

with $\mathbf{Q} = [Q_0, Q_1, \ldots, Q_{m_1}]^{\mathsf{T}}$, $\mathbf{R} = [R_0, R_1, \ldots, R_{m_2}]^{\mathsf{T}}$ and $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_{m_2}]^{\mathsf{T}}$. This also leads to a saddle problem

$$\begin{bmatrix} \mathbf{A}^{\mathsf{T}}\mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{\mathsf{T}}\mathbf{Q} \\ \mathbf{R} \end{bmatrix}, \tag{13}$$

Therefore, we can iteratively solve the saddle point problem in a similar way to that in Sect. 3.2.

At the beginning of the iteration, we specify a set of control points $\{P_{h,l}^{(1,0)}\}_{h=0,l=0}^{n_1,n_2}$, where the first and second superscript $(1, 0)$ of $P_{h,l}$ represent the iteration steps of Uzawa algorithm and LSPIA, respectively, and the initial vector $\lambda_i^{(0)} = [1, 1, 1]^{\mathsf{T}}$ for each interpolation point $R_i$ as the initial value. Then, the initial surface is given by

$$\mathcal{P}^{(1,0)}(u, v) = \sum_{h=0}^{n_1}\sum_{l=0}^{n_2} \mathcal{B}_h(u)\mathcal{B}_l(v)P_{h,l}^{(1,0)}.$$

Suppose that we have the r-th surface defined by $\mathcal{P}^{(\mathrm{r}_1,\mathrm{r})}(u, v)$ in the $\mathrm{r}_1$-th Uzawa iteration, the $j$-th difference vector $\delta_j^{(\mathrm{r}_1,\mathrm{r})}$ between each approximating data point $Q_j$ and the corresponding point on the surface is

$$\delta_j^{(\mathrm{r}_1,\mathrm{r})} = Q_j - \mathcal{P}^{(\mathrm{r}_1,\mathrm{r})}(t_{u_j}, t_{v_j});$$

then, the adjusting vector $\Delta_{h,l}^{(\mathrm{r}_1,\mathrm{r})}$ for the control point with subscript $\{h, l\}$ is defined as

$$\Delta_{h,l}^{(\mathrm{r}_1,\mathrm{r})} = \frac{1}{\mu_1}(\epsilon_{h,l}^{(\mathrm{r}_1,\mathrm{r})} - \theta_{h,l}^{(\mathrm{r}_1)}),$$

where $\mu_1$ is a constant satisfying the condition $\mu_1 > \frac{\gamma_0}{2}$, in which, $\gamma_0$ is the largest eigenvalue of $\mathbf{A}^{\mathsf{T}}\mathbf{A}$, and

$$\epsilon_{h,l}^{(\mathrm{r}_1,\mathrm{r})} = \sum_{j=0}^{m_1} \mathcal{B}_h(t_{u_j})\mathcal{B}_l(t_{v_j})\delta_j^{(\mathrm{r}_1,\mathrm{r})},$$

$$\theta_{h,l}^{(\mathrm{r}_1)} = \sum_{j=0}^{m_2} \mathcal{B}_h(s_{u_j})\mathcal{B}_l(s_{v_j})\lambda_j^{(\mathrm{r}_1-1)}.$$

The control points are then updated as

$$P_{h,l}^{(\mathrm{r}_1,\mathrm{r}+1)} = P_{h,l}^{(\mathrm{r}_1,\mathrm{r})} + \Delta_{h,l}^{(\mathrm{r}_1,\mathrm{r})}, \tag{14}$$

and the $(\mathrm{r}+1)$-th surface is generated by

$$\mathcal{P}^{(\mathrm{r}_1,\mathrm{r}+1)}(u, v) = \sum_{h=0}^{n_1}\sum_{l=0}^{n_2} \mathcal{B}_h(u)\mathcal{B}_l(v)P_{h,l}^{(\mathrm{r}_1,\mathrm{r}+1)}.$$

Same as before, by induction on r, we get a surface sequence defined by $\{\mathcal{P}^{(\mathrm{r}_1,\mathrm{r})}(u, v)\}_{\mathrm{r}=0}^{\infty}$ and its convergence can also be proved in a similar way, as in the curve case. In this way, we can obtain $P_{h,l}^{(\mathrm{r}_1)} = \lim_{\mathrm{r}\to\infty} P_{h,l}^{(\mathrm{r}_1,\mathrm{r})}$ for all pairs $(h, l)$ iteratively.

Once we get those $P_{h,l}^{(\mathrm{r}_1)}$, let $P_{h,l}^{(\mathrm{r}_1+1,0)}$ be $P_{h,l}^{(\mathrm{r}_1)}$, the corresponding surface is defined by $\mathcal{P}^{(\mathrm{r}_1+1,0)}(u, v)$, then

$$\lambda_j^{(\mathrm{r}_1)} = \lambda_j^{(\mathrm{r}_1-1)} + \mu(\mathcal{P}^{(\mathrm{r}_1+1,0)}(s_{u_j}, s_{v_j}) - R_j), \tag{15}$$

where $\mu$ satisfies the condition $\frac{1}{\mu} > \frac{\beta_0}{2}$, in which, $\beta_0$ is the largest eigenvalue of $\mathbf{B}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{B}^{\mathsf{T}}$.

Up to now, we have completed an iteration step of Uzawa algorithm. By induction on $\mathrm{r}_1$, we can get the solution $\mathbf{P}$ in (13). Its convergence can also be proved in a similar way to the curve case.

## 4 Examples and discussions

In this section, we test the CLSPIA algorithm for cubic B-spline curve and surface fitting. In Sect. 4.1, we introduce the details of the parametrization of data points, as well as the selection of the initial control points and knot vectors. Then, we give some examples in Sect. 4.2 and point out the feasibility of fitting a large-scale data set in Sect. 4.3. Finally, a shape preserving fitting method is shown in Sect. 4.4.



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

**Table 1** Resulting interpolation error of different iteration steps for the six curve fitting examples, where $E$ and "error" are the total interpolation error and approximation error after 100,000 iterations, respectively

| Example | $E_0$ | $E_1$ | $E_3$ | $E_7$ | $E$ | Error | References |
|---|---|---|---|---|---|---|---|
| Ex 1 | 1.5258e−05 | 8.1082e−06 | 6.7438e−06 | 4.6532e−06 | 5.2012e−22 | 1.5221e−05 | Fig. 3 |
| Ex 2 | 1.9675e−05 | 6.2619e−06 | 4.8687e−06 | 2.4544e−06 | 1.8556e−21 | 1.2134e−05 | Fig. 4 |
| Ex 3 | 5.451e−05 | 1.3895e−05 | 1.3243e−05 | 1e-05 | 1.7835e−21 | 2.2165e−05 | Fig. 5 |
| Ex 4 | 0.0005 | 0.00055 | 5.1667e−04 | 4.8333e−04 | 2.9683e−21 | 1.9265e−04 | Fig. 6 |
| Ex 5 | 0.00038 | 0.00035 | 0.00036 | 0.00031 | 2.2077e−21 | 4.220e−4 | Fig. 7 |
| Ex 6 | 8.4211e−05 | 6.8421e−05 | 5.2632e−05 | 3.1638e−05 | 7.6653e−22 | 5.0622e−05 | Fig. 2 |

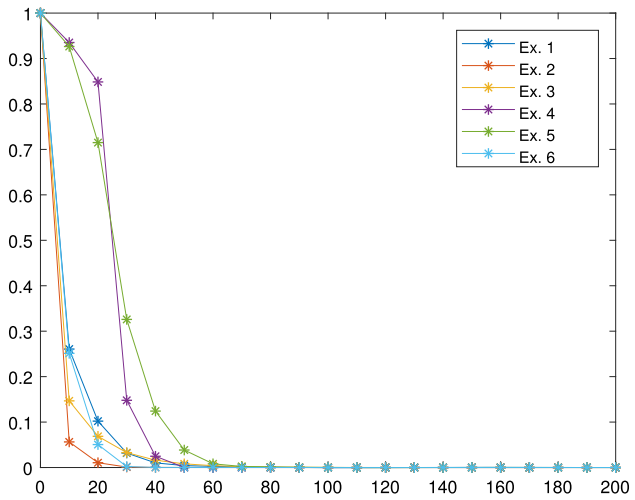



**Fig. 8** Illustration of fitting error with respect to iteration number, using six examples (normalized to [0, 1])

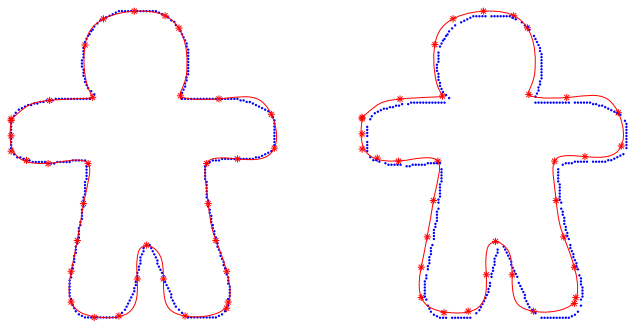



**Fig. 9** When the number of control points is equal to the number of interpolation points, position change of the approximate points has no effect to the resulting curve. (Use the same parameters for both fits.)

### 4.1 Initial parameter selection

In our examples, we use the chord length parametrization method [43] to parametrize data points. Given an ordered set of points $\{K_i\}_{i=0}^m$, the chord length is parametrized to assign parameter $\{\tau_i\}_{i=0}^m$ to $\{K_i\}_{i=0}^m$ as follows:

$$\tau_0 = 0,\ \tau_m = 1,$$
$$\tau_i = \tau_{i-1} + \frac{\|K_i - K_{i-1}\|}{d},\quad i = 1, 2, \ldots, m-1,$$

where $d = \sum_{i=0}^m \|K_i - K_{i-1}\|$ is the total chord length. Chord length parametrization is popular because it is simple to implement, and in general, it leads to fitting curves with nice shape.

The knot vector [43] of the fitting cubic B-spline curve defined by $\mathcal{P}(t) = \sum_{i=0}^n \mathcal{B}_i(t) P_i$ is defined as

$$[0, 0, 0, 0, \overline{\tau}_4, \overline{\tau}_5, \ldots, \overline{\tau}_n, 1, 1, 1, 1],$$

and

$$\overline{\tau}_{j+3} = (1-\alpha)\tau_{i-1} + \alpha\tau_i,\quad j = 1, \ldots, n-3,$$
$$i = \lfloor jd \rfloor,\quad \alpha = jd - i,\quad d = \frac{m+1}{n-2}.$$

Although the initial control points can be chosen arbitrarily in the iterative method, appropriate initial values can reduce the number of iterations. In our experiments, we use the same method as that of [16] to calculate better initial control points because by this method, in general, the initial curve is close to the final fitting curve.

### 4.2 Examples

We present six examples to show the efficiency of the CLSPIA algorithm proposed in Sect. 3.2. The number of control points of the fitting B-spline curves are 55, 30, 20, 37, 22 and 50, respectively. The experimental results are shown in Figs. 2, 3, 4, 5, 6 and 7. In each figure, the initial fitting curve is shown in (a), and the cubic B-spline curves after 1, 3 and 7 iterative steps are shown in (b), (c) and (d), with the final fitted curve shown in (e) and (f). (a–e) show a magnified view of the selected area, which is visible in (f). We use

$$\bar{E}_k = \sum_{j=0}^{m_2} \left\| R_j - \sum_{i=0}^{n} \mathcal{B}_i(s_j) P_i^k \right\|^2$$

to represent the total error, and

$$E_k = \bar{E}_k/(m_2 + 1)$$



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

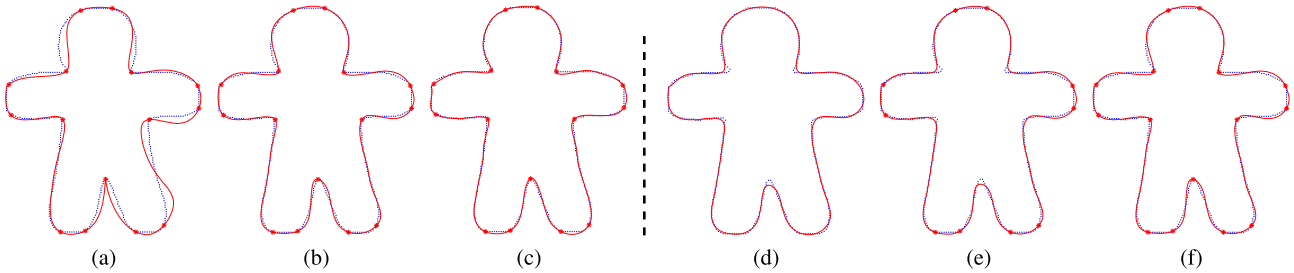



**Fig. 10** A B-spline curve fitting to 370 data points with the different number of control points and the different number of interpolation points. **a**–**c** comparison while interpolating 15 given points with gradually added control points, and the numbers of control points are 25, 35 and 45, respectively. **d**–**f** comparison using 35 control points and with gradually added interpolation points, and the numbers of interpolation points are 0, 10 and 15, respectively.

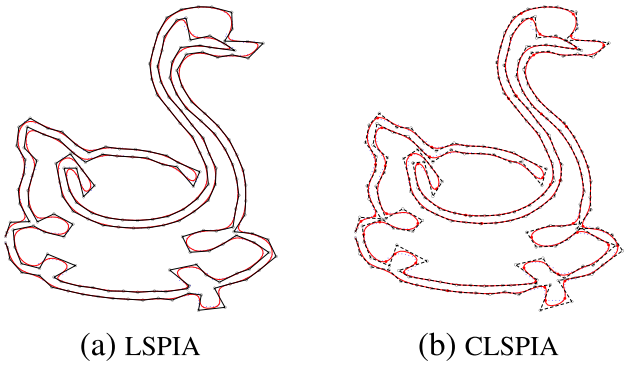



**Fig. 11** Comparison of LSPIA and CLSPIA

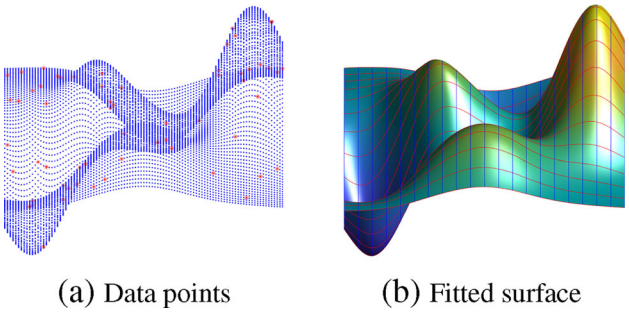



**Fig. 12** A cubic B-spline surface with 13 × 13 control points fitting 10,201 input data points while interpolating 60 given points

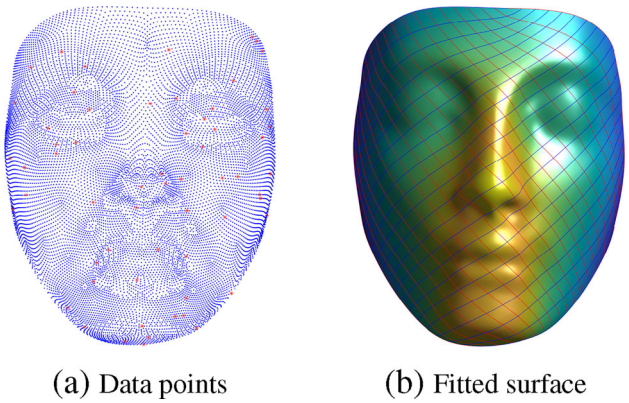



**Fig. 13** A cubic B-spline surface with 35 × 30 control points fitting 12,505 input data points while interpolating 60 given points

to represent the average error of the interpolation points after the $k$-th iteration. Table 1 shows the variation of the error of the interpolation points in the six examples. Figure 8 shows that the convergence speed is faster, and a higher fitting accuracy can be obtained after fewer iterations.

When the number of control points is equal to the number of points to be interpolated, according to (5), the number of equations in this case is equal to that of unknowns, and thus the control points of fitting curve should be the same as an interpolation B-spline curve. We verify this conclusion by an example, in which we randomly select 35 out of 370 points as the set of interpolation points and let the number of control points of the fitting curve be 35. The final fitting curve is shown in Fig. 9a.

Then, we fix the position and parameters of the points to be interpolated and change the positions of the points to be approximated, and the fitting curve is shown in Fig. 9b. From these two figures, we can see that the fitting curves are the same. We also present two examples in which the control points and interpolation points are gradually added, and the results are shown in Fig. 10.

In Fig. 11, we show the results of the LSPIA algorithm [16] and CLSPIA, and the curve generated by CLSPIA can interpolate the given interpolation points (red asterisk points) well, but this will lead to an increase in the fitting error of those points that need to be approximated. In this example, the red solid line is the final fitting curve, the black circles are the control points, the blue solid points are the points that need to be approximated, and the red asterisk points in Fig. 11b are the points that need to be interpolated.

For constrained surface fitting using CLSPIA, we present two examples as shown in Fig. 12 and 13. In the first surface example, there are 10,201 data points, 60 of which need to be interpolated, and a good fitting surface can be obtained with



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

**Table 2** Illustration of the average interpolation error in different iteration steps for the two examples of surface fitting, where $E$ and "error" are the average interpolation error and approximation error after 200,000 iterations, respectively

| Example | $E_0$ | $E_1$ | $E_3$ | $E_7$ | $E_{20}$ | $E$ | Error | References |
|---|---|---|---|---|---|---|---|---|
| Surface 1 | 3.6963e−05 | 4.5548e−06 | 2.7893e−06 | 1.9812e−06 | 5.8185e−07 | 1.6472e−33 | 3.6864e−07 | Fig. 12 |
| Surface 2 | 6.9578e−05 | 6.7447e−05 | 7.5758e−05 | 6.0697e−06 | 3.3012e−06 | 9.9529e−23 | 1.645e−07 | Fig. 13 |

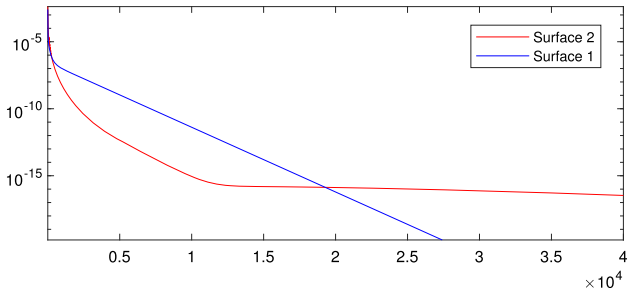



**Fig. 14** In the two examples of surface fitting, we plot the variation of the error of interpolation points with respect to the number of iterations. (Note that the vertical axis scale is logarithmic.)

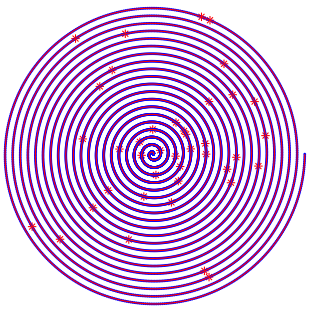


**Fig. 15** A B-spline curve fitting to 10,001 data points

$13 \times 13$ control points. In the second, there are 12,505 data points, 60 of which need to be interpolated, and a good fitting surface can be obtained with $35 \times 30$ control points, and the blue and red solid lines in Figs. 12b and 13b represent the iso-parametric lines in the u-direction and the v-direction, respectively. Table 2 shows that the variation of the average error of the interpolation points in the two examples, and Fig. 14 shows that the convergence speed. As expected, one achieves similar fitting and interpolation properties as that for curve fitting.

### 4.3 Fitting a large-scale data set

When the number of data points is particularly large, the method of directly solving the equations is not feasible, because the condition number of the coefficient matrix is so large and so the equation is relatively ill conditioned [42]. This leads to unstable solutions for solving the equations directly. Another reason is that the matrix is too large and the computational costs become intractable. The iterative method is most suitable for the fitting problem of large-scale data sets. Figure 15 shows an example in which 10,001 data points are approximated and 40 of them are interpolated.

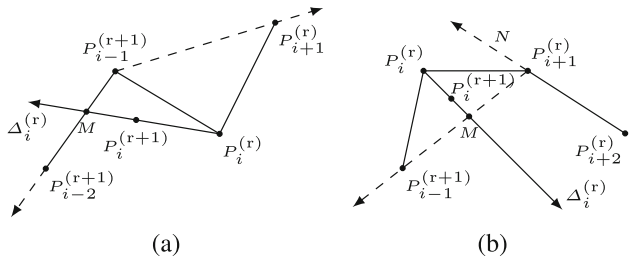



**Fig. 16** Adjust control points according to shape constraints $\{P_i^{(r)}\}$

### 4.4 Shape preserving fitting

In general, shape preserving means that the number of inflection points of the fitted curve is the same as that implied by the sequence of data points. We show that one can easily achieve shape preserving curve fitting using CLSPIA, while it may not be that easy by directly solving a system of linear equations.

First, the initial control points are selected from the data set such that the initial B-spline curve satisfies the shape features implied by the data points. During each iteration, we ensure that the number of inflection points in the updated control polygon $\{P_i^{(r+1)}\}_{i=0}^n$ is the same as that in the control polygon $\{P_i^{(r)}\}_{i=0}^n$. Therefore, the number of inflection points in the limit curve is the same as that in the initial curve. This can be achieved by the following method [16]:

1. For each $i = 0, 1, \ldots, n$, update the control point $P_i^{(r)}$;
2. When updating $P_i^{(r)}$ to $P_i^{(r+1)}$, consider the following two situations:

   (a) Consider four consecutive control points $P_{i-2}^{(r+1)}$, $P_{i-1}^{(r+1)}$, $P_i^{(r)}$ and $P_{i+1}^{(r)}$, if the line segment ending in $P_i^{(r)}$, $P_i^{(r)} + \Delta_i^{(r)}$ intersects with ray from $P_{i-1}^{(r+1)}$ to $P_{i+1}^{(r)}$
   or ray from $P_{i-1}^{(r+1)}$ to $P_{i-2}^{(r+1)}$,
   the intersection is recorded as $M$, then let the new control point $P_i^{(r+1)} := P_i^{(r)} + \alpha(M - P_i^{(r)})$, where $\alpha \in (0, 1)$ (see Fig. 16a).

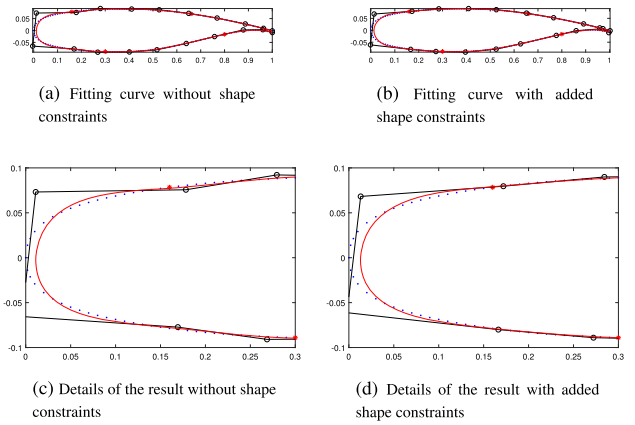


Content courtesy of Springer Nature, terms of use apply. Rights reserved.

(b) Consider four consecutive control points $P_{i-1}^{(r+1)}$, $P_i^{(r)}$, $P_{i+1}^{(r)}$ and $P_{i+2}^{(r)}$, if the line segment ending in $P_i^{(r)}$, $P_i^{(r)} + \Delta_i^{(r)}$ intersects with ray from $P_{i+1}^{(r)}$ to $P_{i-1}^{(r+1)}$ or ray from $P_{i+1}^{(r)}$ to $N$,
the intersection is recorded as $M$, where $N$ is an extension of line from $P_{i+2}^{(r)}$ to $P_{i+1}^{(r)}$.
Then, let the new control point $P_i^{(r+1)} := P_i^{(r)} + \alpha(M - P_i^{(r)})$, where $\alpha \in (0, 1)$ (see Fig. 16b).

3. For $P_1^{(r)}$ and $P_{n-1}^{(r)}$, just consider one of the above conditions.

In our experiment, we let $\alpha = 0.8$, and the experimental results are shown in Fig. 17. The details can be seen in Fig. 17c, d. It can be seen that the number of curve inflection points remains unchanged after adding the shape constraint.



(a) Fitting curve without shape constraints

(b) Fitting curve with added shape constraints

(c) Details of the result without shape constraints

(d) Details of the result with added shape constraints

**Fig. 17** Comparison of fitting curves before and after shape constraints

## 5 Conclusion

We have presented the CLSPIA method for B-spline curve and surface fitting. It is based on LSPIA, the Lagrange multiplier and the Uzawa algorithm. The method aims at finding the best fitting curve approximating the input data set while interpolating some points of the given data. We have proved that the iterative algorithm is convergent, and the limit curve or surface is consistent with the results obtained by constrained least squares fitting. CLSPIA is a typical PIA algorithm, so it has a clear geometric meaning and can dynamically adjust parameters, knot vectors and control points in the iterative process and update the control points in each iteration. CLSPIA is suitable for fitting large-scale data points because it does not need to directly solve a large system of linear equations, and the computational complexity of CLSPIA in one iteration is $O(m)$, where $m$ is the number of points to be fitted. Furthermore, with CLSPIA, we can obtain shape preserving fitting curves by adding a shape constraint to the iterative method. While this paper mainly focuses on CLSPIA for constrained B-spline curve fitting, a brief coverage is also provided to extend the method to constrained surface fitting. As for future work, the method can also be extended for constrained surface fitting while interpolating some of the input data points using other basis functions, such as constrained subdivision surface fitting for complex models of arbitrary topology.

### 5.1 Limitation

For B-spline curve or surface fitting, there are still several parameters that need to be determined, such as the parameterization of data points, the number of control points and the selection of knot vectors. In this paper, we have not discussed how to determine these parameters effectively for practical examples.

**Acknowledgements** This work was supported by the Swiss National Science Foundation (SNF Grant No. 188577), the National Natural Science Foundation of China (Grant No. 61872121) and City University of Hong Kong (SRG Grant No. 7004605).

**Data Availability** The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

## Declarations

**Conflict of interest** We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

## References

1. Axelsson, O.: Iterative Solution Methods. Cambridge University Press, Cambridge (1996)
2. Bai, Z.Z., Wang, Z.Q.: On parameterized inexact Uzawa methods for generalized saddle point problems. Linear Algebra Appl. **428**(11), 2900–2932 (2008)
3. Benzi, M., Golub, G.H., Liesen, J.: Numerical solution of saddle point problems. Acta Numer. **14**, 1–137 (2005)
4. de Boor, C.: A Practical Guide to Splines, vol. 27. Springer, New York (1978)
5. de Boor, C.: How does Agee's smoothing method work. In: Proceedings of the 1979 Army Numerical Analysis and Computers Conference, ARO Report, pp. 79–3 (1979)
6. Chen, J., Wang, G., Jin, C.: Two kinds of generalized progressive iterative approximations. Acta Autom. Sin. **38**(1), 135–138 (2012)
7. Chen, J., Wang, G.J.: Progressive iterative approximation for triangular Bézier surfaces. Comput. Aided Des. **43**(8), 889–895 (2011)
8. Chen, S.: Progressive iterative algorithm for triangular T-Bézier surfaces. Comput. Eng. Appl. **50**(19), 152 (2014)
9. Chen, Z.: Finite Element Methods and Their Applications. Scientific Computation. Springer, Berlin (2006)



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

10. Chen, Z., Luo, X., Tan, L., Ye, B., Chen, J.: Progressive interpolation based on Catmull–Clark subdivision surfaces. Comput. Gr. Forum **27**(7), 1823–1827 (2008)
11. Cheng, F., Fan, F., Huang, C., Wang, J., Lai, S., Miura, K.T.: Smooth surface reconstruction using Doo-Sabin subdivision surfaces. In: 2008 3rd International Conference on Geometric Modeling and Imaging, pp. 27–33 (2008)
12. Cheng, F., Fan, F., Lai, S., Huang, C., Wang, J., Yong, J.: Loop subdivision surface based progressive interpolation. J. Comput. Sci. Technol. **24**(1), 39–46 (2009)
13. Cheng, F.F., Fan, F., Lai, S., Huang, C., Wang, J., Yong, J.: Progressive interpolation using loop subdivision surfaces. In: Chen, F., Jüttler, B. (eds.) Advances in Geometric Modeling and Processing, pp. 526–533. Springer, Berlin (2008)
14. Cheng, X.I.: On the nonlinear inexact Uzawa algorithm for saddle-point problems. SIAM J. Numer. Anal. **37**(6), 1930–1934 (2000)
15. Delgado, J., Peña, J.M.: A comparison of different progressive iteration approximation methods. In: Dæhlen, M., Floater, M., Lyche, T., Merrien, J.L., Mørken, K., Schumaker, L.L. (eds.) Mathematical Methods for Curves and Surfaces, pp. 136–152. Springer, Berlin (2010)
16. Deng, C., Lin, H.: Progressive and iterative approximation for least squares B-spline curve and surface fitting. Comput. Aided Des. **47**, 32–44 (2014)
17. Elman, H.C., Golub, G.H.: Inexact and preconditioned Uzawa algorithms for saddle point problems. SIAM J. Numer. Anal. **31**(6), 1645–1661 (1994)
18. Fan, F., Cheng, F.F., Lai, S.: Subdivision based interpolation with shape control. Comput. Aided Des. Appl. **5**(1–4), 539–547 (2008)
19. Flanigan, F.J., Kazdan, J.L.: Calculus Two: Linear and Nonlinear Functions. Springer, Berlin (1998)
20. Golub, G.H., Van Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore (1983)
21. He, S., Ou, D., Yan, C., Lee, C.H.: A chord error conforming tool path B-spline fitting method for NC machining based on energy minimization and LSPIA. J. Comput. Des. Eng. **2**(4), 218–232 (2015)
22. Hilbert, D., Cohn-Vossen, S.: Geometry and the Imagination. AMS Chelsea Publishing Series. AMS Chelsea Pub, New York City (1999)
23. Ke, Y., Zhu, W., Liu, F., Shi, X.: Constrained fitting for 2D profile-based reverse modeling. Comput. Aided Des. **38**(2), 101–114 (2006)
24. Kineri, Y., Wang, M., Lin, H., Maekawa, T.: B-spline surface fitting by iterative geometric interpolation/approximation algorithms. Comput. Aided Des. **44**(7), 697–708 (2012)
25. Lancaster, P., Šalkauskas, K.: Curve and Surface Fitting: An Introduction. Computational Mathematics and Applications. Academic Press, London (1986)
26. Lin, H.: The convergence of the geometric interpolation algorithm. Comput. Aided Des. **42**(6), 505–508 (2010)
27. Lin, H., Bao, H., Wang, G.: Totally positive bases and progressive iteration approximation. Comput. Math. Appl. **50**(3), 575–586 (2005)
28. Lin, H., Cao, Q., Zhang, X.: The convergence of least-squares progressive iterative approximation with singular iterative matrix. J. Syst. Sci. Complex. **31**, 1618–1632 (2017)
29. Lin, H., Maekawa, T., Deng, C.: Survey on geometric iterative methods and their applications. Comput. Aided Des. **95**, 40–51 (2018)
30. Lin, H., Wang, G., Dong, C.: Constructing iterative non-uniform B-spline curve and surface to fit data points. Sci. China Ser. Inf. Sci. **47**(3), 315–331 (2004)
31. Lin, H., Zhang, Z.: An extended iterative format for the progressive-iteration approximation. Comput. Gr. **35**(5), 967–975 (2011)
32. Lin, Y., Wei, Y.: Fast corrected Uzawa methods for solving symmetric saddle point problems. Calcolo **43**(2), 65–82 (2006)
33. Loucera, C., Iglesias, A., Gálvez, A.: Lévy Flight-Driven Simulated Annealing for B-spline Curve Fitting, pp. 149–169. Springer, Cham (2018)
34. Lu, J.: Convergence analysis of the corrected Uzawa algorithm for symmetric saddle point problems. Appl. Math. A J. Chin. Univers. **29**, 29–35 (2014)
35. Lu, J., Zhang, Z.: A modified nonlinear inexact Uzawa algorithm with a variable relaxation parameter for the stabilized saddle point problem. SIAM J. Matrix Anal. Appl. **31**(4), 1934–1957 (2010)
36. Lu, L.: Weighted progressive iteration approximation and convergence analysis. Comput. Aided Geom. Des. **27**(2), 129–137 (2010)
37. Maekawa, T., Matsumoto, Y., Namiki, K.: Interpolation by geometric algorithm. Comput. Aided Des. **39**(4), 313–323 (2007)
38. Nishiyama, Y., Morioka, M., Maekawa, T.: Loop subdivision surface fitting by geometric algorithms. Poster Proc. Pac. Gr. **2008**, 67–74 (2008)
39. Pang, H., Li, W.: A corrected Uzawa method for symmetric saddle point problems. Math. Numer. Sin. **31**, 231 (2009)
40. Park, H.: An error-bounded approximate method for representing planar curves in B-splines. Comput. Aided Geom. Des. **21**(5), 479–497 (2004)
41. Park, H., Lee, J.H.: B-spline curve fitting based on adaptive curve refinement using dominant points. Comput. Aided Des. **39**(6), 439–451 (2007)
42. Pereyra, V., Scherer, G.: Large scale least squares scattered data fitting. Appl. Numer. Math. **44**(1), 225–239 (2003)
43. Piegl, L., Tiller, W.: The NURBS Book, 2nd edn. Springer, Berlin (1997)
44. Qi, D., Tian, Z., Zhang, Y., Zheng, J.: The method of numeric polish in curve fitting. Acta Math. Sin. **18**(3), 173–184 (1975)
45. Rogers, D., Fog, N.: Constrained B-spline curve and surface fitting. Comput. Aided Des. **21**(10), 641–648 (1989)
46. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia (2003)
47. Shi, L., Wang, R.: An iterative algorithm of nurbs interpolation and approximation. J. Math. Res. Expos. **26**, 735–743 (2006)
48. Smith, R.E., Jr., Price, J.M., Howser, L.M.: A Smoothing Algorithm Using Cubic Spline Functions. National Aeronautics and Space Administration, Washington (1974)
49. Uyar, K., Ülker, E.: B-spline curve fitting with invasive weed optimization. Appl. Math. Model. **52**, 320–340 (2017)
50. Uzawa, H.: Iterative methods for concave programming. Stud. Linear Nonlinear Program. **6**, 154–165 (1958)
51. Vandergraft, J.S.: Chapter 4 - interpolation and approximation. In: Vandergraft, J.S. (ed.) Introduction to Numerical Computations, 2nd edn., pp. 89–138. Academic Press, London (1983)
52. Xiong, Y., Li, G., Mao, A.: Convergence analysis for B-spline geometric interpolation. Comput. Gr. **36**(7), 884–891 (2012)
53. Yamaguchi, F.: A design method of free form surfaces by a computer display (1st report). J. Jpn. Soc. Precis. Eng. **43**(506), 168–173 (1977)




Content courtesy of Springer Nature, terms of use apply. Rights reserved.

54. Zhang, L., Ge, X., Tan, J.: Least square geometric iterative fitting method for generalized B-spline curves with two different kinds of weights. Vis. Comput. **32**(9), 1109–1120 (2016)
55. Zhang, L., Li, Y., Yang, Y., Tan, J.: Generalized progressive iterative approximation for Said-Ball bases on triangular domains. J. Image Gr. **19**(2), 275–282 (2014)
56. Zhang, L., et al.: The iteration method for sample-based polynomial approximation of rational B-spline curves. J. Inf. Comput. Sci. **12**(3), 865–872 (2015)
57. Zhao, Y., Lin, H.: The PIA property of low degree non-uniform triangular B-B patches. In: 2011 12th International Conference on Computer-Aided Design and Computer Graphics, pp. 239–243 (2011)
58. Zheng, W., Bo, P., Liu, Y., Wang, W.: Fast B-spline curve fitting by L-BFGS. Comput. Aided Geom. Des. **29**(7), 448–462 (2012)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

**Qingjun Chang** is currently a Ph.D. candidate in the Faculty of Informatics, Università della Svizzera italiana, Switzerland. He received his M.Sc. degree from Hangzhou Dianzi University in 2020, and B.Sc. degree from Ningbo Institute of Technology, Zhejiang University in 2017. His research interests include geometry processing, computer-aided geometric design and computer graphics.

**Weiyin Ma** is an associate professor from the Department of Mechanical Engineering (MNE) at City University of Hong Kong (CityU), Hong Kong, China. He received his B.Sc. and M.Sc. degrees from East China Institute of Technology (ECIT) in 1982 and 1985, respectively, and M.Eng. and Ph.D. degrees from Katholieke Universiteit Leuven (K.U.Leuven) in 1989 and 1994, respectively. His present research interests include digital geometry processing, computer-aided geometric design, CAD/CAM/CAE, isogeometric analysis and simulation, 3D printing and rapid manufacturing.

**Chongyang Deng** is now a full professor in the School of Sciences, Hangzhou Dianzi University, China. He obtained a B.Sc. degree in applied mathematics from Jilin University and a Ph.D. degree in CAGD and Computer graphics from Zhejiang University in 1997 and 2008, respectively. His research interests include spans of computer-aided geometric design and computer graphics.



Content courtesy of Springer Nature, terms of use apply. Rights reserved.

# Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com