Jacobi–PIA Algorithm for Bi-Cubic B-Spline Interpolation Surfaces

Chengzhi Liu
School of Mathematics and Finance, Hunan University of Humanities, Science and Technology
Dixing Road, Loudi City, Hunan Province, P.R. China
it-rocket@163.com

Juncheng Li
School of Mathematics and Finance, Hunan University of Humanities, Science and Technology
Dixing Road, Loudi City, Hunan Province, P.R. China
lijuncheng82@126.com

Lijuan Hu
School of Mathematics and Finance, Hunan University of Humanities, Science and Technology
Dixing Road, Loudi City, Hunan Province, P.R. China
1023236794@qq.com

Abstract

Based on the Jacobi splitting of collocation matrices, we in this paper exploited the Jacobi–PIA format for bi-cubic B-spline surfaces. We first present the Jacobi–PIA scheme in term of matrix product, which has higher computational efficiency than that in term of matrix-vector product. To analyze the convergence of Jacobi–PIA, we transform the matrix product iterative scheme into the equivalent matrix-vector product scheme by using the properties of the Kronecker product. We showed that with the optimal relaxation factor, the Jacobi–PIA format for bi-cubic B-spline surface converges to the interpolation surface. Numerical results also demonstrated the effectiveness of the proposed method.

1. Introduction

Consider the interpolation problem by using the classic bi-cubic B-spline surfaces. Given an ordered data-set $\{\boldsymbol{p}_{ij}\}_{i=1,\cdots,m}^{j=1,\cdots,n}$ in $\mathbb{R}^3$, each point $\boldsymbol{p}_{ij}$ is assigned a pair of parameters (u_i, v_j), where $u_1 = v_1 = 0$ and

$$\begin{cases} u_i = u_{i-1} + \dfrac{1}{n}\displaystyle\sum_{j=1}^{n} \|\boldsymbol{p}_{ij} - \boldsymbol{p}_{i-1,j}\|,\ i = 2, 3, \cdots, m; \\ v_j = v_{j-1} + \dfrac{1}{m}\displaystyle\sum_{i=1}^{m} \|\boldsymbol{p}_{ij} - \boldsymbol{p}_{i,j-1}\|,\ j = 2, 3, \cdots, n. \end{cases}$$

We want to find a bi-cubic B-spline surface $\boldsymbol{S}^*(u, v)$ that interpolates $\{\boldsymbol{p}_{ij}\}_{i=1,\cdots,m}^{j=1,\cdots,n}$, *i.e.*,

$$\boldsymbol{S}^*(u_i, v_j) = \boldsymbol{p}_{ij}, \quad i = 1, \cdots, m;\ j = 1, \cdots, n. \tag{1}$$

Date interpolation arises in a wide range of applications throughout computational science and engineering. Oftentimes, the interpolation curves/surfaces can be obtained by solving linear systems of equations. For a small number of data, direct solvers to linear systems are the preferred choices and are widely used because they are more reliable. While for large-scale data, direct solvers are always unstable and time-consuming, hence it requires efficient algorithms for solving large linear systems. It is well known that for large-scale systems, iterative methods are primary options and a number of iterative

methods have emerged. For example, iterative methods based on splittings of coefficient matrices, subspace iterative methods, preconditioning iterative methods, and so on.

In recent years an iterative method, namely progressive iteration approximation (PIA), with clear geometric meaning was proposed. Due to its clear geometric meaning and stable convergence, the PIA gradually attracts researchers' attention and plays a significant role in data interpolation [9, 8]. Unfortunately, such a class of PIAs converges slowly and thus affects the computational efficiency of data interpolation, especially on problems with large-scale data to be interpolated. To overcome this disadvantage, numerous accelerating techniques and alternatives have been proposed in the literature, see [10, 7, 4, 12, 13, 21, 11, 2, 6, 14, 20, 5, 15].

Regardless of its geometric significance, the PIA is mathematically equivalent to the Richardson iterative method ([2]), which is a classic splitting method for solving linear systems. As known to all, the convergence rate of splitting iterative methods mainly depends on their spectral radii of iteration matrices. And the smaller the spectral radius is, the faster the splitting method converges. Therefore, an appropriate splitting method will make the spectral radius of the iterative matrix smaller and result in a faster convergence rate, *e.g.*, the modified Richardson splitting iteration [1], also known as the weighted PIA [6]; the Jacobi splitting iteration (Jacobi–PIA) [14]; the Gauss-Seidel splitting iteration (GS-PIA) [14, 20]; the Hermitian and skew-Hermitian splitting iteration [5]. The mentioned methods have faster convergence rates than the classical Richardson method/PIA.

We notice in [14] that Liu *et al*. only discuss the Jacobi-PIA for cubic B-spline curves not discuss the iteration for surfaces. Therefore, we in this paper exploit the Jacobi-PIA for bi-cubic B-spline surfaces based on the Jacobi splitting and analyze its convergence. The remainder of this paper is organized as follows: Some related works, as well as the PIA for bi-cubic B-spline surfaces, are described in Section 2. The Jacobi-PIA for bi-cubic B-spline surfaces is exploited in Section 3, and its convergence is analyzed in Section 4. Some numerical examples are given in Section 5. Finally, we draw our conclusions in Section 6.

2. Related work

2.1. Some conclusions of matrices

For convenience, the identity matrix of order n is denoted by I_n; the spectra and the spectral radius of a matrix are denoted by $\lambda_i(\cdot)$ and $\rho(\cdot)$, respectively; the minimal and the maximal eigenvalue of a matrix are denoted by $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$, respectively; the column-stacking vector of a matrix is denoted by $\mathbf{vec}(\cdot)$, e.g.,

$$\mathbf{vec}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right) = [\ 1 \quad 4 \quad 7 \quad 2 \quad 5 \quad 8 \quad 3 \quad 6 \quad 9\ ]^T.$$

Definition 1 ([18]) *Let $A = (a_{ij}) \in \mathbb{R}^{n\times n}$, $B = (b_{ij}) \in \mathbb{R}^{m\times m}$. Then the Kronecker product of A and B is defined as*

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{bmatrix}.$$

Lemma 1 ([18]) *The properties of the Kronecker product are given as follows:*

(1) $\mathbf{vec}(AXB) = \left(B^T \otimes A\right)\mathbf{vec}(X)$, *where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$ and $X \in \mathbb{R}^{n\times m}$;*

(2) *$A \otimes B$ is invertible if and only if both A and B are invertible, and $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$;*

(3) *If A, B, C and D are matrices of such size that one can form the matrix products AC and BD, then $(A\otimes B)(C \otimes D) = (AC) \otimes (BD)$.*

(4) *Let $\lambda_1, \ldots, \lambda_n$ and $\mu_1, \ldots, \mu_m$ be the eigenvalues of $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$, respectively. Then the eigenvalues of $A \otimes B$ are $\lambda_i\mu_j, i = 1, 2, \cdots n, j = 1, 2, \cdots, m$.*

2.2. Matrix representation of bi-cubic B-spline interpolation surface

We expand the interpolation question mentioned in the introduction. Firstly, we construct the knot vectors $\boldsymbol{U} = \{u_i\}_{i=-2}^{m+3}$ and $\boldsymbol{V} = \{v_j\}_{j=-2}^{n+3}$ based on the parameters $\{u_i\}_{i=1}^{m}$ and $\{v_j\}_{j=1}^{n}$, respectively. To ensure that the bi-cubic B-spline surface interpolates at the boundary points, we adopt the multiple-knot method, *i.e.*,

$$u_{-2} = \ldots = u_1, u_m = \ldots = u_{m+3}, v_{-2} = \ldots = v_1, v_n = \ldots = v_{n+3}.$$

Let $N_i^3(u)$ and $N_j^3(v)$ be the cubic B-spline basic functions defined on the knot vectors $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. Then the bi-cubic B-spline interpolation surface in (1) can be expressed as

$$\boldsymbol{S}^*(u,v) = \sum_{i=-2}^{m-1}\sum_{j=-2}^{n-1} \boldsymbol{p}_{i+2,j+2}^* N_i^3(u) N_j^3(v), \quad (u,v) \in [u_1, u_m] \otimes [v_1, v_n],$$

where $\left\{\boldsymbol{p}_{ij}^*\right\}_{i=1,\cdots,m}^{j=1,\cdots,n}$ are the control points need to be determined (Since the multiple-knot method is adopted, the boundary control points are determined.).

Letting $\boldsymbol{P}^* = \left(\boldsymbol{p}_{ij}^*\right)_{i=1,\cdots,m}^{j=1,\cdots,n}$, $\boldsymbol{P} = \left(\boldsymbol{p}_{ij}\right)_{i=1,\cdots,m}^{j=1,\cdots,n}$. Then the formula (1) can be written in the following matrix equation

$$B_1 \boldsymbol{P}^* B_2^T = \boldsymbol{P}, \tag{2}$$

where

$$B_1 = \begin{bmatrix} 1 & 0 & & & \\ N_{-1}^3(u_2) & N_0^3(u_2) & N_1^3(u_2) & & \\ & \ddots & \ddots & \ddots & \\ & & N_{m-4}^3(u_{m-1}) & N_{m-3}^3(u_{m-1}) & N_{m-2}^3(u_{m-1}) \\ & & & 0 & 1 \end{bmatrix}$$

and

$$B_2 = \begin{bmatrix} 1 & 0 & & & \\ N_{-1}^3(v_2) & N_0^3(v_2) & N_1^3(v_2) & & \\ & \ddots & \ddots & \ddots & \\ & & N_{n-4}^3(v_{n-1}) & N_{n-3}^3(v_{n-1}) & N_{n-2}^3(v_{n-1}) \\ & & & 0 & 1 \end{bmatrix}$$

are the so-called collocation matrices.

According to Lemma 1, one can alternatively solve instead of (2) the equivalent the matrix equation

$$(B_2 \otimes B_1)\mathbf{vec}(\boldsymbol{P}^*) = \mathbf{vec}(\boldsymbol{P}). \tag{3}$$

Remark 1 *According to Lemma 1, it is easy to verify that the solution to (3) is the column-stacking vector of the one to (2). We also note in [19, 16] that the system (3) is suitable for theoretical analysis but impractical to solve its numerical solution because the system (3) is more ill-conditioned and costly than the matrix equation (2). Therefore, we in this paper exploit the approximate interpolation scheme in the matrix form, then transform it into its equivalent matrix-vector product scheme to analyze its convergence.*

2.3. PIA for bi-cubic B-spline surfaces

In [7], the PIA was proposed to solve the aforementioned interpolation problem. For the data points $\left\{\boldsymbol{p}_{ij}\right\}_{i=1,\cdots,m}^{j=1,\cdots,n} \in \mathbb{R}^3$ to be interpolated and the knot vectors $\boldsymbol{U} = \{u_i\}_{i=-2}^{m+3}$, $\boldsymbol{V} = \{v_j\}_{j=-2}^{n+3}$. Let

$$\begin{cases} \boldsymbol{p}_{ij}^{(0)} = \boldsymbol{p}_{ij}, i = 1, \ldots, m; j = 1, \ldots, n; \\ \boldsymbol{p}_{0j}^{(0)} = \boldsymbol{p}_{1j}^{(0)}, \boldsymbol{p}_{m+1,j}^{(0)} = \boldsymbol{p}_{m,j}^{(0)}, j = 1, \ldots, n; \\ \boldsymbol{p}_{i0}^{(0)} = \boldsymbol{p}_{i1}^{(0)}, \boldsymbol{p}_{i,n+1}^{(0)} = \boldsymbol{p}_{i,n}^{(0)}, i = 1, \ldots, m; \\ \boldsymbol{p}_{00}^{(0)} = \boldsymbol{p}_{11}^{(0)}, \boldsymbol{p}_{0,n+1}^{(0)} = \boldsymbol{p}_{1n}^{(0)}, \boldsymbol{p}_{m+1,0}^{(0)} = \boldsymbol{p}_{m,1}^{(0)}, \boldsymbol{p}_{m+1,n+1}^{(0)} = \boldsymbol{p}_{m,n}^{(0)}. \end{cases}$$

Then we can obtain a bi-cubic B-spline surface

$$\boldsymbol{S}^{(0)}(u,v)=\sum_{i=-2}^{m-1}\sum_{j=-2}^{n-1}\boldsymbol{p}_{i+2,j+2}^{(0)}N_i^3(u)N_j^3(v),\quad (u,v)\in[u_1,u_m]\otimes[v_1,v_n],$$

that approximately interpolates $\{\boldsymbol{p}_{ij}\}_{i=1,\cdots,m}^{j=1,\cdots,n}$.

Let $\boldsymbol{\delta}_{ij}^{(0)}=\boldsymbol{p}_{ij}-\boldsymbol{S}^{(0)}(u_i,v_j), i=1,\ldots,m; j=1,\ldots,n$ be the difference vector between the points $\boldsymbol{p}_{ij}$ and their approximate interpolation surface $\boldsymbol{S}^{(0)}(u,v)$. Then we can update the control points

$$\begin{cases}\boldsymbol{p}_{ij}^{(1)}=\boldsymbol{p}_{ij}^{(0)}+\boldsymbol{\delta}_{ij}^{(0)}, & \text{if } i\in\{1,\ldots,m\} \text{ and } j\in\{1,\ldots,n\};\\ \boldsymbol{p}_{ij}^{(1)}=\boldsymbol{p}_{ij}^{(0)}, & \text{if } i\in\{0,m+1\} \text{ or } j\in\{0,n+1\},\end{cases}$$

and generate the approximate interpolation surface after one iteration

$$\boldsymbol{S}^{(1)}(u,v)=\sum_{i=-2}^{m-1}\sum_{j=-2}^{n-1}\boldsymbol{p}_{i+2,j+2}^{(1)}N_i^3(u)N_j^3(v),\quad (u,v)\in[u_1,u_m]\otimes[v_1,v_n].$$

Similarly, let $\boldsymbol{S}^{(k)}(u,v)$ be the approximate interpolation surface after k iterations, and let $\boldsymbol{\delta}_{ij}^{(k)}=\boldsymbol{p}_{ij}-\boldsymbol{S}^{(k)}(u_i,v_j), i=1,\ldots,m; j=1,\ldots,n$ be the difference vector. Then the $(k+1)$-th approximate interpolation surface can be defined by

$$\boldsymbol{S}^{(k+1)}(u,v)=\sum_{i=-2}^{m-1}\sum_{j=-2}^{n-1}\boldsymbol{p}_{i+2,j+2}^{(k+1)}N_i^3(u)N_j^3(v),\quad (u,v)\in[u_1,u_m]\otimes[v_1,v_n],$$

where

$$\begin{cases}\boldsymbol{p}_{ij}^{(k+1)}=\boldsymbol{p}_{ij}^{(k)}+\boldsymbol{\delta}_{ij}^{(k)}, & \text{if } i\in\{1,\ldots,m\} \text{ and } j\in\{1,\ldots,n\};\\ \boldsymbol{p}_{ij}^{(k+1)}=\boldsymbol{p}_{ij}^{(k)}, & \text{if } i\in\{0,m+1\} \text{ or } j\in\{0,n+1\}.\end{cases}\tag{4}$$

Thus we obtain a sequence of bi-cubic B-spline surfaces $\left\{\boldsymbol{S}^{(k)}(u,v)\right\}_{k=0}^{\infty}$ that approximately interpolate $\{\boldsymbol{p}_{ij}\}_{i=1,\cdots,m}^{j=1,\cdots,n}$. The method to generate $\left\{\boldsymbol{S}^{(k)}(u,v)\right\}_{k=0}^{\infty}$ is known as the PIA for bi-cubic B-spline surfaces [7].

3. Jacobi–PIA for bi-cubic B-spline surfaces

For the aforementioned data $\{\boldsymbol{p}_{ij}\}_{i=1,\cdots,m}^{j=1,\cdots,n}\in\mathbb{R}^3$, knot vectors $\boldsymbol{U}=\{u_i\}_{i=-2}^{m+3}$, $\boldsymbol{V}=\{v_j\}_{j=-2}^{n+3}$ and initial control points $\left\{\boldsymbol{p}_{ij}^{(0)}\right\}_{i=0,\cdots,m+1}^{j=0,\cdots,n+1}$, we can iteratively generate a sequence of bi-cubic B-spline surfaces

$$\boldsymbol{S}^{(k)}(u,v)=\sum_{i=-2}^{m-1}\sum_{j=-2}^{n-1}\boldsymbol{p}_{i+2,j+2}^{(k)}N_i^3(u)N_j^3(v),(u,v)\in[u_1,u_m]\otimes[v_1,v_n],\ k=0,1,\ldots,\tag{5}$$

that approximately interpolate $\{\boldsymbol{p}_{ij}\}_{i=1,\cdots,m}^{j=1,\cdots,n}$. Here the control points in (5) are defined by

$$\begin{cases}\boldsymbol{p}_{ij}^{(k)}=\boldsymbol{p}_{ij}^{(k-1)}+\omega\frac{1}{N_{i-2}^3(u_i)N_{j-2}^3(v_j)}\boldsymbol{\delta}_{ij}^{(k-1)}, & \text{if } i\in\{1,\ldots,m\} \text{ and } j\in\{1,\ldots,n\};\\ \boldsymbol{p}_{ij}^{(k)}=\boldsymbol{p}_{ij}^{(k-1)}, & \text{if } i\in\{0,m+1\} \text{ or } j\in\{0,n+1\},\end{cases}\tag{6}$$

where ω is a nonnegative real number. Different from the method in (4), we in (6) multiply the difference vector $\boldsymbol{\delta}_{ij}^{(k)}$ by $\omega\frac{1}{N_{i-2}^3(u_i)N_{j-2}^3(v_j)}$ when we updating the control points of bi-cubic B-spline surfaces.

Letting $\boldsymbol{P}^{(k)}=\left(\boldsymbol{p}_{ij}^{(k)}\right)_{i=1,\cdots,m}^{j=1,\cdots,n}, k=0,1,\ldots$. Then (6) can be written in matrix form

$$\boldsymbol{P}^{(k+1)}=\boldsymbol{P}^{(k)}+\omega D_1^{-1}(\boldsymbol{P}-B_1\boldsymbol{P}^{(k)}B_2^T)D_2^{-T},\quad k=0,1,\ldots.\tag{7}$$

In [19], Tian *et al*. proposed the Jacobi iterative method for matrix equations. Let $B_1 = D_1 - (L_1 + U_1)$ and $B_2 = D_2 - (L_2 + U_2)$ be the Jacobi splitting of the collocation matrices B_1 and B_2, respectively. Here the matrices D_i, L_i and U_i are the diagonal parts, strictly lower triangular parts and strictly upper triangular parts of $B_i (i = 1, 2)$, respectively. Then the Jacobi iterative format for solving (2) is

$$\boldsymbol{P}^{(k+1)} = \boldsymbol{P}^{(k)} + D_1^{-1} \left(\boldsymbol{P} - B_1 \boldsymbol{P}^{(k)} B_2^T \right) D_2^{-T}. \tag{8}$$

Remark 2 *Compared with the iteration (8), we in (7) multiply a relaxation factor ω to guarantee its convergence. Although Liu* et al. *have shown that the Jacobi–PIA for cubic B-spline curves is convergent [14], it's not always true for the Jacobi–PIA for bi-cubic B-spline surfaces, which will be verified by an numerical example later. Therefore, we introduce a relaxation factor ω and then determine the optimal relaxation factor to guarantee the convergence of the proposed iterative method.*

The bi-cubic B-spline surface is said to have the Jacobi–PIA property if the limit of the sequence (5) is the interpolation surface of $\{\boldsymbol{p}_{ij}\}_{i=1,\cdots,n}^{j=1,\cdots,m}$, *i.e.*,

$$\lim_{k\to\infty} \boldsymbol{S}^{(k)}(u_i, v_j) = \boldsymbol{S}^*(u_i, v_j) = \boldsymbol{p}_{ij}.$$

The Jacobi–PIA property also indicates that the sequence of control points converges to the control points of the interpolation surface $\boldsymbol{S}^*(u, v)$, i.e.,

$$\lim_{k\to\infty} \boldsymbol{p}_{ij}^{(k)} = \boldsymbol{p}_{ij}^*, \quad i = 1, \ldots, m; j = 1, \ldots, n.$$

We refer to the iteration (7) as the Jacobi–PIA format.

According to Lemma 1, the iteration (7) can be expressed as follows

$$\mathbf{vec}\left(\boldsymbol{P}^{(k+1)}\right) = \left(I_{m\times n} - \omega \left(D_2^{-1} \otimes D_1^{-1}\right)(B_2 \otimes B_1)\right) \mathbf{vec}\left(\boldsymbol{P}^{(k)}\right) + \omega \left(D_2^{-1} \otimes D_1^{-1}\right) \mathbf{vec}(\boldsymbol{P}). \tag{9}$$

Hence the convergence of (7) can be alternatively transformed into that of (9).

Remark 3 *From (9), D_1 and D_2 are diagonal matrices composed of the main diagonals of B_1 and B_2, respectively. Hence both the diagonal entries of D_1 and D_2 are different, so is the matrix $\omega \left(D_2^{-1} \otimes D_1^{-1}\right)$. On the other hand, we note in [21] that Zhang* et al. *proposed the PIA with different weights (DWPIA) for tensor product surfaces. It can be expressed as*

$$\mathbf{vec}\left(\boldsymbol{P}^{(k+1)}\right) = (I_{m\times n} - W(B_2 \otimes B_1)) \mathbf{vec}\left(\boldsymbol{P}^{(k)}\right) + W\mathbf{vec}(\boldsymbol{P}),$$

where W is a diagonal matrix with different diagonal entries, i.e., the so-called different weights in [21]. Since W and $\omega \left(D_2^{-1} \otimes D_1^{-1}\right)$ are diagonal matrices with different diagonal entries, the proposed Jacobi–PIA can also seem as a variant of DWPIA. Clearly, the choices of the weights in DWPIA will influence the convergence of the DWPIA. However, we notice in [21] that they do not give a strategy to determine the optimal weights. The analytic determination of optimal weights appears to be daunting. In the next section, we present the optimal relaxation factor ω for Jacobi–PIA, which also gives analytic optimal weights for DWPIA alternatively.

4. Convergence analysis of Jacobi–PIA

Definition 2 ([3]) *A matrix is said to be* totally nonnegative *if all the minors of A are nonnegative;* totally positive *if all the minors of A are positive.*

Lemma 2 *For $i = 1, 2$, let D_i be the diagonal matrix composed of the main diagonal of the collocation matrix B_i. Then the diagonal matrices D_1^{-1} and D_2^{-1} are totally nonnegative.*

Proof 1 *Note in [7] that the diagonals of the collocation matrices B_1 and B_2 are positive. Hence, D_1 and D_2 are diagonal matrices with positive diagonal entries. It follows from Definition 2 that a diagonal matrix is totally nonnegative if all the diagonal entries are positive. This shows that D_1^{-1} and D_2^{-1} are totally nonnegative.*

Lemma 3 ([3]) *Totally nonnegative matrices are closed under multiplication; all the eigenvalues of a totally nonnegative matrix are nonnegative real numbers.*

Lemma 4 ([17]) *Let $\boldsymbol{x}^{(k+1)} = M\boldsymbol{x}^{(k)} + \boldsymbol{c}$ be any iteration for solving a linear system $A\boldsymbol{x} = \boldsymbol{b}$. Then the iterative sequence $\left\{\boldsymbol{x}^{(k)}\right\}_{k=0}^{\infty}$ converges to the solution of $A\boldsymbol{x} = \boldsymbol{b}$ for all starting vectors $\boldsymbol{x}^{(0)}$ and for all $\boldsymbol{b}$ if and only if the spectral radius of the iteration matrix is less than* 1, i.e., *$\rho(M) < 1$.*

Lemma 5 ([17]) *Given a linear system of equations $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^{n\times 1}$. Let $\boldsymbol{x}^{(k+1)} = (I_n - \alpha A)\boldsymbol{x}^{(k)} + \alpha\boldsymbol{b}$ be the Richardson iterative method for solving $A\boldsymbol{x} = \boldsymbol{b}$, where α is a nonnegative real number. Then the Richardson iterative method has the fastest convergence rate if*

$$\alpha = \frac{2}{\lambda_{\max}(A) + \lambda_{\min}(A)}.$$

And in this case, the spectral radius of the iteration matrix $I_n - \alpha A$ if

$$\rho(I_n - \alpha A) = \frac{\lambda_{\max}(A) - \lambda_{\min}(A)}{\lambda_{\max}(A) + \lambda_{\min}(A)}.$$

Theorem 1 *Let $\boldsymbol{S}^{(k)}(u, v)$ be the sequence of bi-cubic B-spline surfaces defined as in (5). Then the bi-cubic B-spline surface has the property of Jacobi–PIA if*

$$\omega = \frac{2}{\lambda_{\max}(D_1^{-1}B_1)\lambda_{\max}(D_2^{-1}B_2) + \lambda_{\min}(D_1^{-1}B_1)\lambda_{\min}(D_2^{-1}B_2)}. \tag{10}$$

And in this case, the Jacobi–PIA has the fastest convergence rate.

Proof 2 *Note that the Jacobi–PIA can also be seem as the Richardson iterative method for solving*

$$\left(D_2^{-1} \otimes D_1^{-1}\right)(B_2 \otimes B_1)\mathbf{vec}\left(\boldsymbol{P}^*\right) = \left(D_2^{-1} \otimes D_1^{-1}\right)\mathbf{vec}\left(\boldsymbol{P}\right).$$

It follows from Lemmata 1 and 5 that the Jacobi–PIA has the fastest convergence rate if

$$\begin{aligned}\omega &= \frac{2}{\lambda_{\max}\left(\left(D_2^{-1} \otimes D_1^{-1}\right)(B_2 \otimes B_1)\right) + \lambda_{\min}\left(\left(D_2^{-1} \otimes D_1^{-1}\right)(B_2 \otimes B_1)\right)} \\ &= \frac{2}{\lambda_{\max}(D_2^{-1}B_2)\lambda_{\max}(D_1^{-1}B_1) + \lambda_{\min}(D_2^{-1}B_2)\lambda_{\min}(D_1^{-1}B_1)}.\end{aligned}$$

And in this case, the spectral radius of the iteration matrix $I_{m\times n} - \omega\left(D_2^{-1} \otimes D_1^{-1}\right)(B_2 \otimes B_1)$ is

$$\rho = \frac{\lambda_{\max}(D_1^{-1}B_1)\lambda_{\max}(D_2^{-1}B_2) - \lambda_{\min}(D_1^{-1}B_1)\lambda_{\min}(D_2^{-1}B_2)}{\lambda_{\max}(D_1^{-1}B_1)\lambda_{\max}(D_2^{-1}B_2) + \lambda_{\min}(D_1^{-1}B_1)\lambda_{\min}(D_2^{-1}B_2)}.$$

From Lemmata 2 and 3, both $D_1^{-1}B_1$ and $D_2^{-1}B_2$ are totally nonnegative, hence $\lambda_i(D_1^{-1}B_1) \geq 0$ and $\lambda_i(D_2^{-1}B_2) \geq 0$. It is clear that with the optimal ω,

$$\rho\left(I_{m\times n} - \omega\left(D_2^{-1} \otimes D_1^{-1}\right)(B_2 \otimes B_1)\right) < 1.$$

It follows from Lemma 4 that the iterative sequence (9) is convergent. By the equivalence of (7) and (9), we can conclude that the bi-cubic B-spline surface has the property of Jacobi–PIA.

5. Numerical results

In this section, we present several numerical examples to assess the effectiveness of the proposed algorithm. For comparison, we also tested the PIA ([7]) and the DWPIA [21]. All numerical experiments were performed in Matlab R2016a.

For approximating interpolation, the interpolation error of the kth interpolation surface $\boldsymbol{S}^{(k)}(u, v)$ is defined by

$$\epsilon^{(k)} = \max_{\substack{1\leq i\leq m\\ 1\leq j\leq n}} \left\|\boldsymbol{p}_{ij} - \boldsymbol{S}^{(k)}(u_i, v_j)\right\|,$$

where $\|\cdot\|$ denotes the Euclidean norm.

Example 1 *Consider interpolating* 20 *points:* $(1,1,1)$, $(1,2,4)$, $(1,3,2)$, $(1,4,2)$, $(1,5,2)$, $(2,1,2)$, $(2,2,2)$, $(2,3,3)$, $(2,4,6)$, $(2,5,4)$, $(3,1,3)$, $(3,2,2)$, $(3,3,4)$, $(3,4,4)$, $(3,5,3)$, $(4,1,4)$, $(4,2,6)$, $(4,3,1)$, $(4,4,1)$, $(4,5,2)$.

Example 2 *Consider interpolating* 40×40 *points sampled equidistantly from the shell surface*

$$\begin{cases} x = \frac{1}{5}\left(1-\frac{v}{2\pi}\right)\cos(2v)(1+\cos u)+\frac{1}{10}\cos(2u) \\ y = \frac{1}{5}\left(1-\frac{v}{2\pi}\right)\sin(2v)(1+\cos u)+\frac{1}{10}\sin(2u) \\ z = \frac{v}{2\pi}+\frac{1}{5}\left(1-\frac{v}{2\pi}\right)\sin(2u) \end{cases} ; \quad 0 \le u, v \le 2\pi.$$

Example 3 *Consider interpolating* 10×10 *points sampled from* $f(x,y) = sin(\sqrt{x^2+y^2})/\sqrt{x^2+y^2}$ *in the following way*

$$\left(-8+\frac{16(i-1)}{9}, -8+\frac{16(j-1)}{9}, f\left(-8+\frac{16(i-1)}{9}, -8+\frac{16(j-1)}{9}\right)\right), i = 1,2,\ldots,10; j = 1,2,\ldots,10.$$

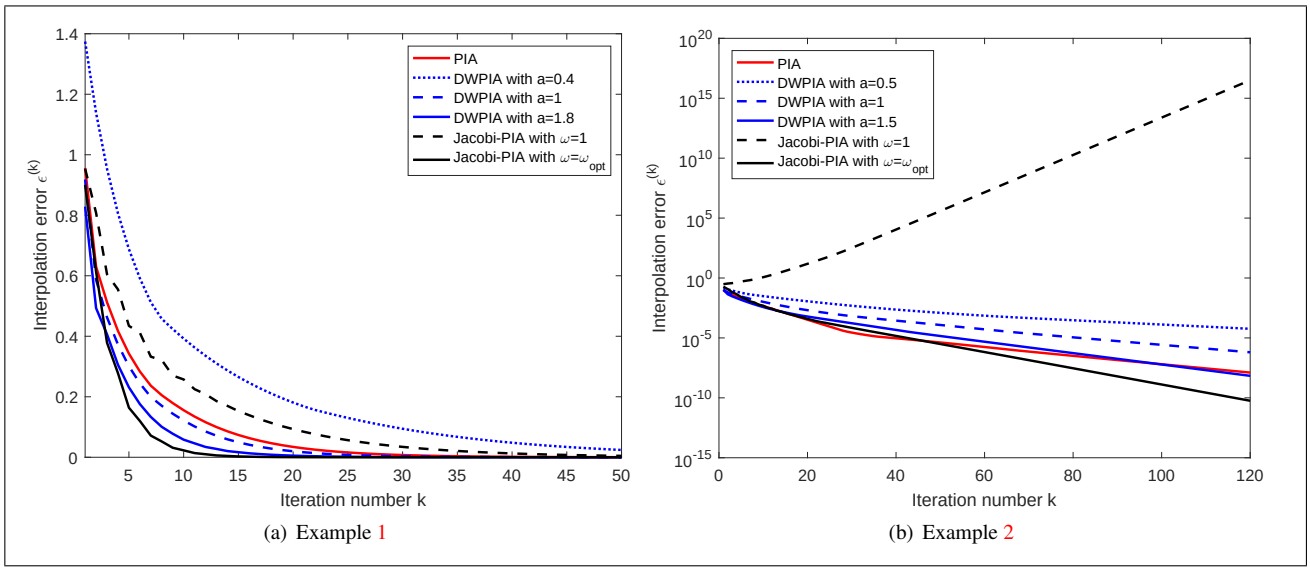
Example 4 *Consider interpolating* $m \times n$ *points sampled from*

$$g(x,y) = \frac{3}{4}e^{-\frac{(9x-2)^2+(9y-2)^2}{4}} + \frac{3}{4}e^{-\frac{(9x+1)^2}{49}-\frac{9y+1}{10}} + \frac{1}{2}e^{-\frac{(9x-7)^2+(9y-3)^2}{4}} - \frac{1}{5}e^{-(9x-4)^2-(9y-7)^2}$$

in the following way

$$\left(\frac{i-1}{m-1}, \frac{j-1}{n-1}, g\left(\frac{i-1}{m-1}, \frac{j-1}{n-1}\right)\right), \quad i = 1,2,\ldots,m; j = 1,2,\ldots,n.$$

We begin with the experiments by employing the PIA, the DWPIA, and the Jacobi–PIA to interpolate the points given in Examples 1 and 2. In our experiments, we tested the DWPIA with different parameters a and tested the Jacobi-PIA with different parameters ω. In Figure 1, we show the interpolation errors after k iterations, where x-axis represents the number of iterations, y-axis represents the interpolation error. To make the results distinct, the y-axis in Figure 1(b) is plotted on a logarithmic scale. As shown in Figure 1, for the optimal parameter ω (denoted by ω_{opt}), the interpolation error of Jacobi–PIA converges fastest as the number of iterations increases. It is also evident from Figure 1(b) that the Jacobi–PIA with $\omega = 1$ is divergent. This indicates that the Jacobi–PIA with $w = \omega_{opt}$ converges, but may diverge with $w = 1$, which confirms the conclusion in Remark 2. On the other hand, even though the DWPIA does converge for different parameters a, the convergence rate also varies for different a. In other words, the convergence rate of DWPIA is even slower than that of PIA if the value of a is not appropriate. This allows us to find an appropriate a by testing the DWPIA with different parameters a. Consequently, it will affect the computational efficiency of the DWPIA.



(a) Example 1 (b) Example 2

Figure 1. Interpolation errors curves of PIA, DWPIA and Jacobi–PIA in Examples 1 and 2.

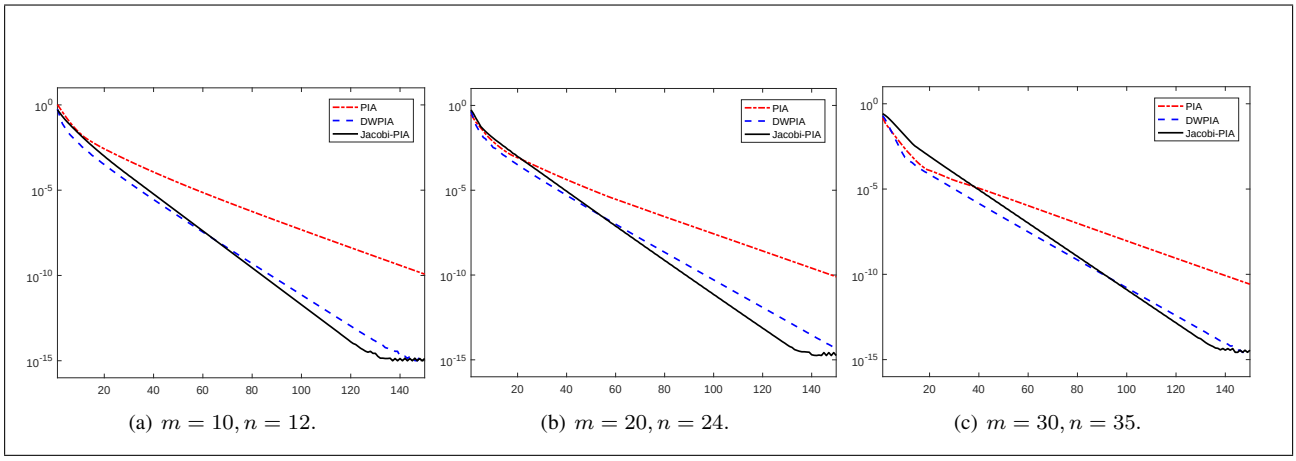
In the following experiments, we took $\omega = \omega_{opt}$ when we employed the Jacobi–PIA to interpolate the points given in Examples 1 – 4. And when we employed the DWPIA to test Examples 1 - 4, we took $a = 1.7, 1.5, 1.8$ and 1.6, respectively. When we tested Example 4, we took different $m \times n$: were 10×12, 20×24 and 30×35.

The spectral radii of iteration matrices of the PIA, the DWPIA, and the Jacobi–PIA are listed in Table 1. It is evident from Table 1 that the spectral radii of the Jacobi–PIA are less than those of the PIA and the DWPIA, hence the Jacobi–PIA would have the fastest convergence rate. It should be pointed out that when we tested Example 3, the spectral radius of the Jacobi–PIA with $w = 1$ is 1.2340, while the spectral radius of the Jacobi–PIA with the optimal w is 0.7734, which also confirms the conclusion in Remark 2.

Table 1. Spectral radii of PIA, DWPIA and Jacobi–PIA in Examples 1 – 4.

Method	Example 1	Example 2	Example 3	Example 4		
				$m = 10, n = 12$	$m = 20, n = 24$	$m = 30, n = 35$
PIA	0.8586	0.8882	0.8794	0.8885	0.8929	0.8929
DWPIA	0.7178	0.8168	0.7788	0.7941	0.8071	0.8093
Jacobi–PIA	0.6645	0.7984	0.7734	0.7796	0.7984	0.8003

By implementing the PIA, the DWPIA, and the Jacobi–PIA, we list in Table 2 the interpolation errors of Example 3 and show in Figures 2 the interpolation errors of Example 4. In Figure 2, the x-axis represents the number of iterations, y-axis represents the interpolation error and is plotted on a logarithmic scale. As the results reported in Table 2 and Figure 2 show, the interpolation errors obtained by the Jacobi–PIA are close to those obtained by the PIA and the DWPIA at the first iterations and decrease even faster than those obtained by the PIA and the DWPIA as the number of iterations increases. The interpolation error obtained by the Jacobi–PIA reaches the machine precision (near 10^{-15}) first.



(a) $m = 10, n = 12$. (b) $m = 20, n = 24$. (c) $m = 30, n = 35$.

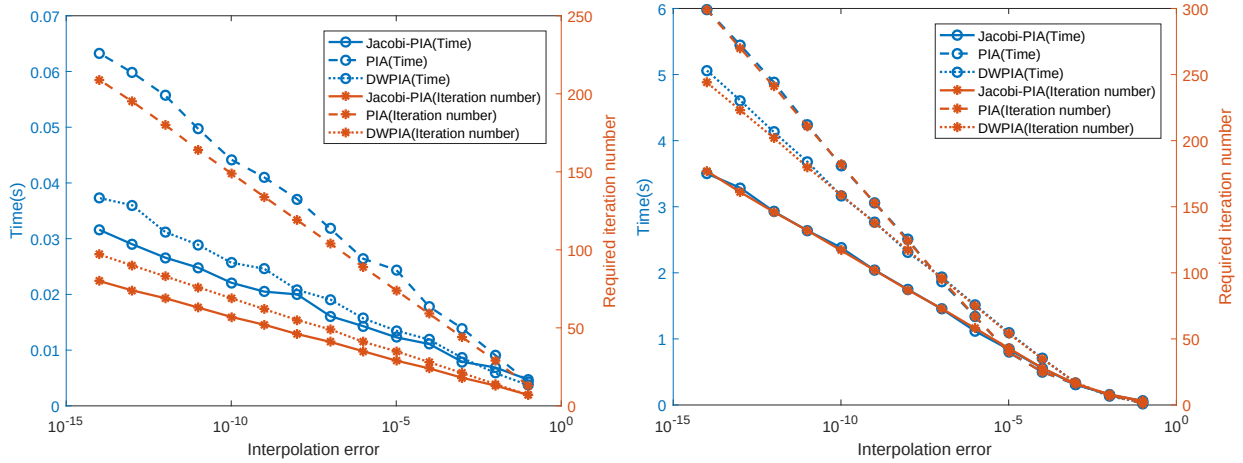
Figure 2. Interpolation errors of PIA, DWPIA and Jacobi–PIA in Example 4.

Under the requirement of the same interpolation error, we show in Figure 3 the computational time (in second) and the required number of iterations by employing the PIA, the DWPIA, and the Jacobi–PIA to test Examples 1 – 4. We took

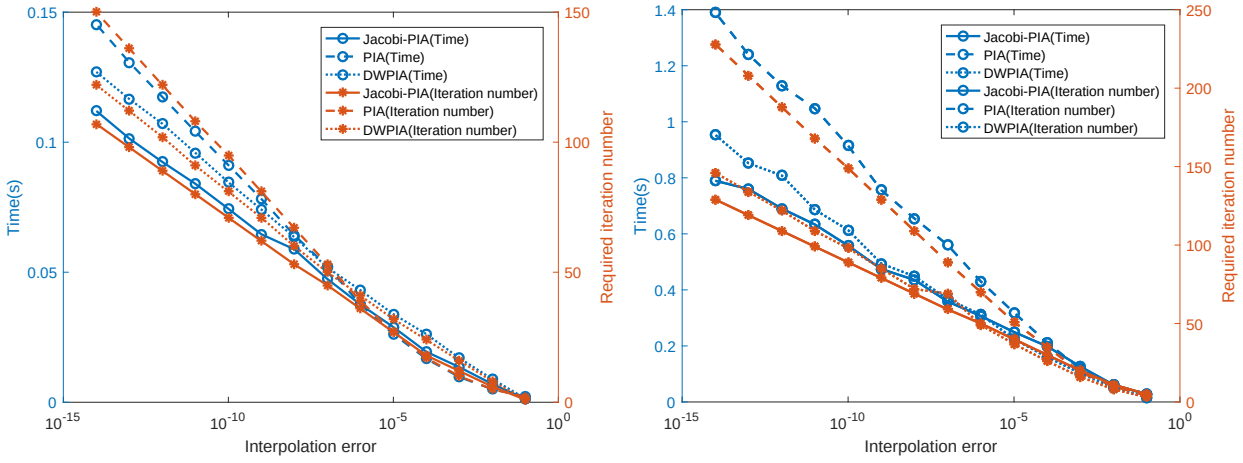
k	PIA	DWPIA	Jacobi–PIA
1	9.01×10^{-2}	8.27×10^{-2}	7.96×10^{-2}
2	4.35×10^{-2}	4.80×10^{-2}	3.86×10^{-2}
5	6.97×10^{-3}	1.76×10^{-2}	1.07×10^{-2}
10	6.45×10^{-4}	3.90×10^{-3}	1.53×10^{-3}
20	9.35×10^{-6}	2.11×10^{-4}	3.56×10^{-5}
50	1.53×10^{-8}	3.96×10^{-8}	1.82×10^{-9}
80	1.05×10^{-10}	2.49×10^{-11}	8.09×10^{-13}

Table 2. Interpolation errors of PIA, DWPIA and Jacobi–PIA in Example 3.

$m = 20$ and $n = 24$ when we tested Example 4. In our experiments, the required interpolation errors were set to 10^{-1}, 10^{-2}, $\ldots$, 10^{-14}, respectively. Figure 3 is plotted with two y-axes, which represent the computational time (left) and the required number of iterations (right), respectively. The x-axis is plotted on a logarithmic scale and represents the required interpolation error. From Figure 3, we can observe that the computational time is coincident with the required number of iterations. The computational time depends mainly on the requirement of the interpolation error. More importantly, the computational time of the Jacobi–PIA is less than those of the other methods. This benefits from the fact that the Jacobi–PIA has the smallest number of iterations for the same interpolation error.



(a) Example 1. (b) Example 2.



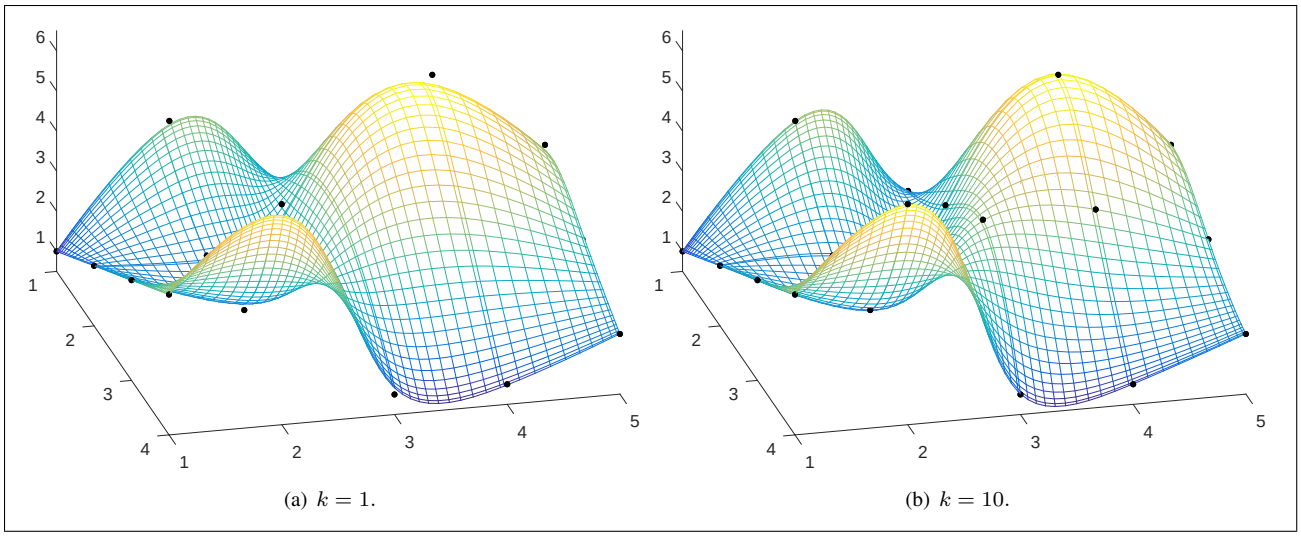
(c) Example 3. (d) Example 4.

Figure 3. Runtime (in second) and required number of iterations vs interpolation error in Examples 1-4.

By employing the Jacobi–PIA to interpolate the points given in Examples 1 – 4, we obtain the bi-cubic B-spline interpolation surfaces, which are shown in Figures 4 – 7, respectively. The interpolation surfaces in Figure 7 are obtained to interpolate 20×24 points sampled from $g(x, y)$ given in Example 4. In Figures 4 – 7, all subfigures (a) are obtained by performing one Jacobi–PIA iteration, and subfigures (b) are obtained by performing 10 Jacobi–PIA iterations. As the results reported in Figures 4 – 7 show, the interpolation surfaces obtained by the Jacobi–PIA have good performances in approximating interpolation.

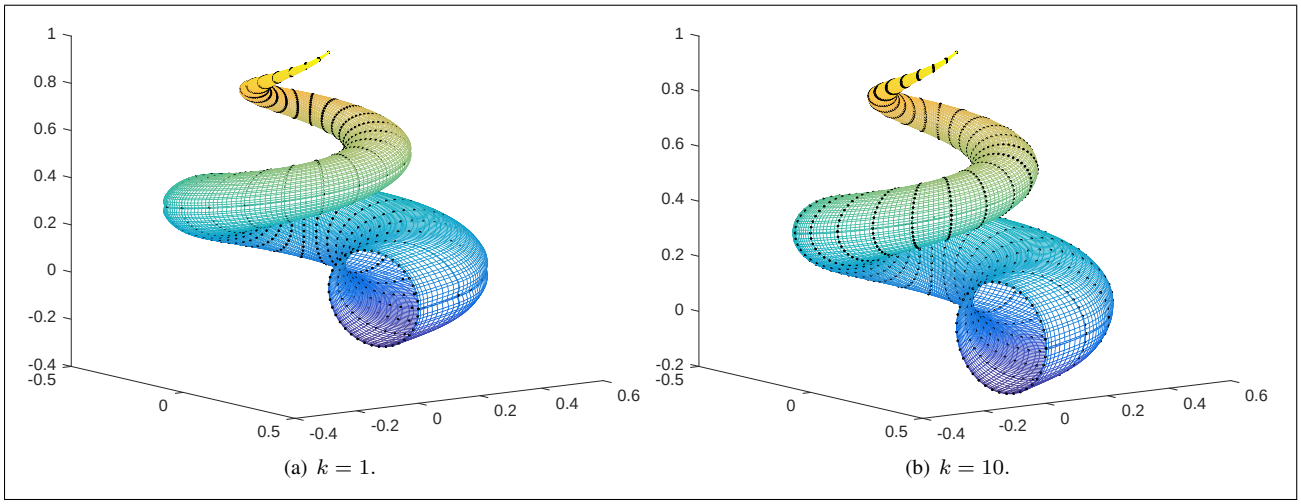
6. Conclusions

In this paper, we have exploited the Jacobi–PIA for tensor product bi-cubic B-spline surfaces. By introducing a relaxation factor, the Jacobi–PIA can also be seen as a variant of the DWPIA and has the fastest convergence rate with the optimal relaxation factor. Thus it provides a reference strategy for determining the analytic optimal weights of DWPIA. Our numerical



(a) $k = 1$. (b) $k = 10$.

Figure 4. Bi-cubic B-spline interpolation surfaces obtained by performing Jacobi–PIA k iterations in Example 1.



(a) $k = 1$. (b) $k = 10$.

Figure 5. Bi-cubic B-spline interpolation surfaces obtained by performing Jacobi–PIA k iterations in Example 2.

results show that the Jacobi–PIA performs well in data interpolation.

Acknowledgments

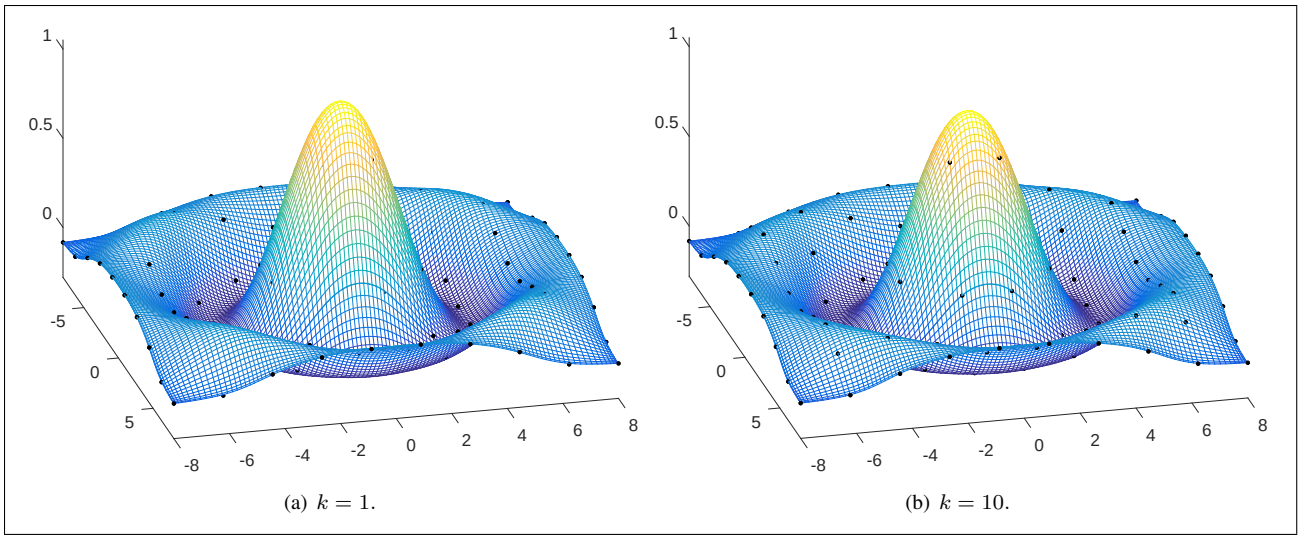
This work was supported by the National Natural Science Foundation of China [grant numbers 12101225]; the Natural Science Foundation of Hunan Province [grant number 2020JJ5267, 2021JJ30373]; and the Scientific Research Funds of Hunan Provincial Education Department [grant number 21B0790].

References

[1] J. Carnicer, J. Delgado, and J. Peña. Richardson method and totally nonnegative linear systems. *Linear Algebra and its Applications*, 11:2010–2017, 2010. 2

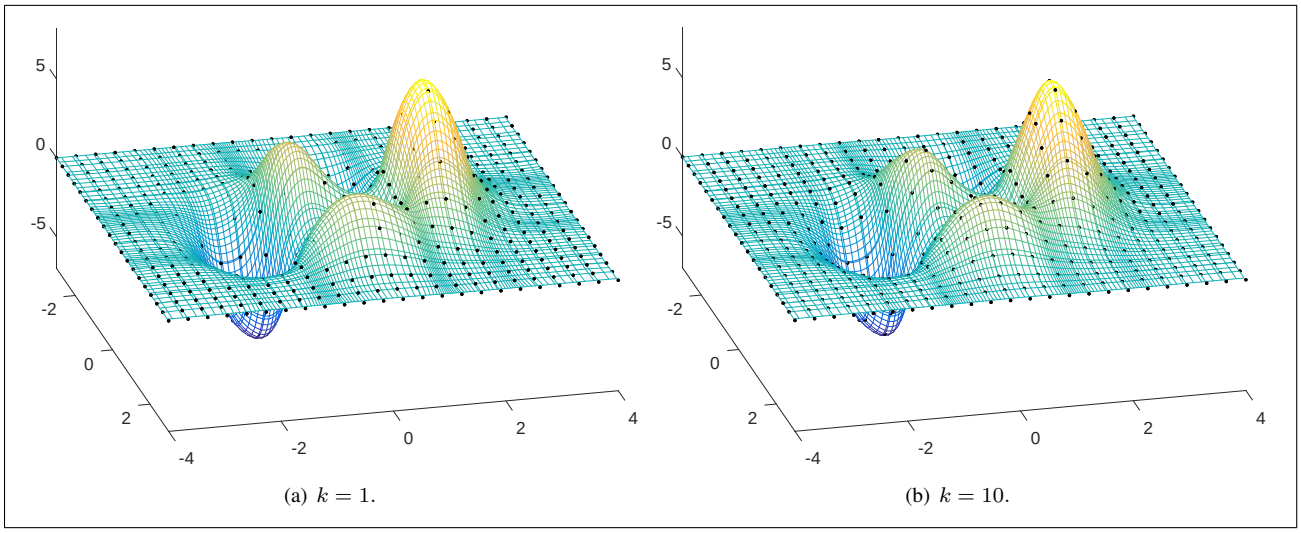
[2] J. Carnicer, J. Delgado, and J. Peña. On the progressive iterative approximation property and alternative iterations. *Computer Aided Geometric Design*, 28:523–526, 2011. 2

[3] S. M. Fallat and C. R. Johnson. *Totally nonnegative matrices, Princeton Series in Applied Mathematics*. Princeton University Press, 2011. 5



(a) $k = 1$. (b) $k = 10$.

Figure 6. Bi-cubic B-spline interpolation surfaces obtained by performing Jacobi–PIA k iterations in Example 3.



(a) $k = 1$. (b) $k = 10$.

Figure 7. Bi-cubic B-spline interpolation surfaces obtained by performing Jacobi–PIA k iterations in Example 4.

[4] Y. F. Hamza, H. Lin, and Z. Li. Implicit progressive-iterative approximation for curve and surface reconstruction. *Computer Aided Geometric Design*, 77:101817, 2019. 2

[5] H. Hu, L.and Shou and Z. Dai. Hss–iteration–based iterative interpolation of curves and surfaces with ntp bases. *Wireless Networks*, 27:3523–3535, 2021. 2

[6] L. L. Weighted progressive iterative approximation and convergence analysis. *Computer Aided Geometric Design*, 2:129–137, 2010. 2

[7] H. Lin, Q. Cao, and Z. X. The convergence of least-squares progressive iterative approximation for singular least-squares fitting system. *Journal of Systems Science & Complexity*, 31(6):1618–1632, 2018. 2, 3, 4, 5, 6

[8] H. Lin, T. Maekawa, and C. Deng. Survey on geometric iterative methods and their applications. *Computer-Aided Design*, 95:40–51, 2018. 2

[9] H. Lin, G. Wang, and C. Dong. Constructing iterative non-uniform b-spline curve and surface to fit data points. *Science in China (Series E)*, 47(3):315–331, 2004. 2

[10] H. Lin and Z. Zhang. An efficient method for fitting large data sets using T-splines. *SIAM Journal on Scientific Computing*, 35(6):A3052–A3068, 2013. 2

[11] C. Liu, X. Han, and L. J. Progressive-iterative approximation by extension of cubic uniform b-spline curves. *Journal of Computer-Aided Design & Computer Graphics*, 31(6):899–910, 2019. 2

[12] C. Liu and Z. Liu. Implicit progressive-iterative approximation for curve and surface reconstruction. *Mathematics*, 8:1503, 2020. 2

[13] C. Liu, Z. Liu, and X. Han. Preconditioned progressive iterative approximation for tensor product bzier patches. *Mathematics and Computers in Simulation*, 185:372–383, 2021. 2

[14] X. Liu and D. C. Jacobi-pia algorithm for non-uniform cubic b-spline curve interpolation. *Journal of Computer-Aided Design & Computer Graphics*, 27:485–491, 2015. 2, 5

[15] X. Liu and D. C. Non-uniform cubic b-spline curve interpolation algorithm of gs–pia. *Journal of Hangzhou Dianzi University*, 2:79–82, 2015. 2

[16] Z. Liu, Z. Li, C. Ferreira, and Y. Zhang. Stationary splitting iterative methods for the matrix equation $AXB = C$. *Applied Mathematics and Computation*, 378:125195, 2020. 3

[17] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003. 6

[18] V. Simoncini. Computational methods for linear matrix equations. *SIAM Review*, 58:377–441, 2016. 2

[19] Z. Tian. The Jacobi and Gauss-Seidel-type iteration methods for the matrix equation $AXB = C$. *Applied Mathematics and Computation*, 292:63–75, 2017. 3, 5

[20] Z. Wang, Y. Li, and C. Deng. Convergence proof of gs–pia algorithm. *Journal of Computer-Aided Design & Computer Graphics*, 30:60–66, 2018. 2

[21] L. Zhang, L. Zhao, and J. Tan. Progressive iterative approximation with different weights and its application. *Journal of Zhejiang University (Science Edition)*, 41(1):22–27, 2017. 2, 5, 6