THE WEIGHTED GS-PIA ALGORITHM FOR CUBIC B-SPLINE CURVE INTERPOLATIONS AND CONVERGENCE ANALYSIS

ZHONGYUN LIU*, JIAN YANG*, XIAOFEI XU*, MENGZHU LIN*, AND YULIN ZHANG†

Abstract. The weighted Gauss-Seidel-progressive iterative approximation (WGS-PIA) algorithm for cubic B-spline curve interpolations is considered in this paper. The convergence of the WGS-PIA algorithm is analyzed, and an upper bound which is strictly smaller than one for the contraction factor of this WGS-PIA algorithm is derived. It is shown that for cubic B-spline curve interpolations, the GS-PIA algorithm converges faster than the Jacobi-PIA (J-PIA) algorithm, and that there always exists a positive weight ω such that the WGS-PIA converges faster than GS-PIA. Particularly, we derive a formula for the effective weight $\omega^\star$ and the "theoretically optimal" weight ω_m, which significantly improves the performance of the WGS-PIA algorithm with minimal additional cost. The numerical experiments are shown that for a given termination tolerance, the number of iterations and the CPU time required by the WGS-PIA algorithm are less than those required by the GS-PIA algorithm.

Key words. curve interpolations; WGS-PIA algorithm; convergence; cubic B-spline basis; optimal weight

AMS subject classifications. 15A48, 15A51, 65D17, 65F10

1. Introduction. The progressive iterative approximation (PIA) method [10] (also called the geometric iterative method [11]), plays an important role in curve and surface-fitting for a given set of data points. This method is of clear geometric meaning, stable convergence and simple iterative scheme. Above all, the PIA avoids solving a system of linear equations directly, which may cause numerical instability when the number of given data points becomes larger. For those reasons, the PIA method and its variants have intrigued researchers for decades. In particular, the PIA method has been widely applied in CAGD, data fitting and reverse engineering, mesh generation and so on [9, 11].

Given an ordered set of data points $\{\mathbf{q}_l\}_{l=0}^n$ in $\mathbb{R}^2$ or $\mathbb{R}^3$, each point $\mathbf{q}_l$ is assigned a parameter value t_l for all $l = 0, 1, \cdots, n$, and they follow

$$t_0 = 0,\ t_l = \sum_{i=0}^{l-1} \|\mathbf{q}_{i+1} - \mathbf{q}_i\|_2, l = 1, \cdots, n.$$

It is clear that $t_0 < t_1 < \cdots < t_n$. We then define

$$t_{-3} = t_{-2} = t_{-1} = t_0,\ t_n = t_{n+1} = t_{n+2} = t_{n+3}$$

to get $\{t_l\}_{l=-3}^{n+3}$, generating the cubic B-spline basis $\{b_l(t)\}_{l=0}^n$ by the de Boor-Cox formula [21]. These functions are a set of locally effective piecewise polynomial functions, used to construct complex smooth curves and surfaces [10, 12, 14, 18].

Throughout this paper, all PIA-type algorithms are assumed to be cubic B-spline curve interpolation algorithms if not specifically mentioned. The classical PIA process for cubic B-spline curve interpolations is as follows.

The PIA scheme. Setting $\{\mathbf{p}_l^{(0)}\}_{l=0}^n = \{\mathbf{q}_l\}_{l=0}^n$ (*the initial control polygon*) and constructing $\mathbf{c}^{(0)}(t) = \sum_{l=0}^n \mathbf{p}_l^{(0)} b_l(t)$ (*the initial interpolating curve*), for $k = 1, 2, \cdots$, until $\{\mathbf{p}_l^{(k)}\}_{l=0}^n$ converges,

*School of Mathematics and Statistics, Changsha University of Science and Technology, Changsha 410076, P. R. China (Liu: liuzhongyun@263.net; Yang: yangjian000731@163.com; Xu: 674204659@qq.com; Lin: 2503571138@qq.com)

†Centro de Matemática, Universidade do Minho, 4710-057 Braga, Portugal (Yulin Zhang: zhang@math.uminho.pt).

- to compute $\boldsymbol{\delta}_l^{(k)} = \mathbf{q}_l - \mathbf{c}^{(k-1)}(t_l), \quad \text{for } l = 0, 1, \cdots, n;$
- to update

$$\mathbf{p}_l^{(k)} = \mathbf{p}_l^{(k-1)} + \boldsymbol{\delta}_l^{(k)}, \quad \text{for } l = 0, 1, \cdots, n; \tag{1.1}$$

- to construct the k-th interpolating curve $\mathbf{c}^{(k)}(t) = \sum_{l=0}^{n} \mathbf{p}_l^{(k)} b_l(t)$.

Eq. (1.1) generates the k-th control polygon $\{\mathbf{p}_l^{(k)}\}_{l=0}^{n}$ and satisfies the following relation

$$\mathbf{p}_l^{(k)} = \mathbf{p}_l^{(k-1)} + \mathbf{q}_l - \sum_{j=0}^{n} \mathbf{p}_j^{(k-1)} b_j(t_l), \quad \text{for } l = 0, 1, \cdots, n. \tag{1.2}$$

Denote $\mathbf{P}^{(k)} = [\mathbf{p}_0^{(k)}, \mathbf{p}_1^{(k)}, \cdots, \mathbf{p}_n^{(k)}]^T, \mathbf{Q} = [\mathbf{q}_0, \mathbf{q}_1, \cdots, \mathbf{q}_n]^T$; then, in matrix-matrix form, Eq. (1.2) can be expressed as follows,

$$\mathbf{P}^{(k)} = \mathbf{P}^{(k-1)} + (\mathbf{Q} - B\mathbf{P}^{(k-1)}), \quad \text{for} \quad k = 1, 2, \cdots, \tag{1.3}$$

where

$$B = \begin{bmatrix} b_0(t_0) & b_1(t_0) & \cdots & b_n(t_0) \\ b_0(t_1) & b_1(t_1) & \cdots & b_0(t_0) \\ \vdots & \vdots & & \vdots \\ b_0(t_n) & b_1(t_n) & \cdots & b_n(t_n) \end{bmatrix} \tag{1.4}$$

is the collocation matrix resulting from the basis functions $\{b_l(t)\}_{l=0}^{n}$ at $t_j, j = 0, 1, \cdots, n$. Very clearly, the PIA (1.3) is a special Richardson iteration for matrix equation $B\mathbf{P} = \mathbf{Q}$, see, for instance, references [3, 16].

To accelerate the convergence rate of PIA, some variants of PIA were proposed. Those variants can be written as the following form

$$\mathbf{P}^{(k)} = \mathbf{P}^{(k-1)} + M^{-1}(\mathbf{Q} - B\mathbf{P}^{(k-1)}). \tag{1.5}$$

From the viewpoint of numerical linear algebra, Eq. (1.5) can be viewed as a classical splitting iteration for matrix equation $B\mathbf{P} = \mathbf{Q}$, with the splitting defined as $B = M - N$. Usually, the matrices M, $M^{-1}B$ and N are called preconditioning, preconditioned and residual matrices, respectively, see, for example, the reference [2]. So we refer to Eq. (1.5) as the preconditioned PIA (PPIA). For example, the PPIA (1.5) becomes

- the PIA, if $M = I$;
- the WPIA [15], if $M = \frac{1}{\omega} I$;
- the J-PIA [14], if $M = D$;
- the GS-PIA [18], if $M = D + L$;
- the PIA with different weights (DWPIA) [20], if $M = diag(\omega_1, \cdots, \omega_{n+1})$,

where

$$B = D + L + U \tag{1.6}$$

with L, D, U denoting strictly lower triangular part, diagonal part, strictly upper triangular part of the collocation matrix B (1.4), respectively.

In this paper, we focus on the acceleration of the GS-PIA algorithm for cubic B-spline curve interpolations [18]. Motivated by the work for the WPIA algorithm in [15], we consider the WGS-PIA algorithm for cubic B-spline curve interpolations. Additionally, we offer a more succinct proof of convergence for both GS-PIA

and J-PIA compared to the analyses presented in [14, 18]. Notably, we demonstrate that GS-PIA exhibits faster convergence rates compared to J-PIA.

The organization of this paper is as follows: after recalling some basic definitions and known results in next section, we introduce the WGS-PIA algorithm and analyze its convergence in Section 3, a highly efficient weight selection strategy is introduced and some numerical examples are shown in Section 4, and a brief conclusion is followed in last section.

Notation. The notations $|K|$, $\|K\|$, $\rho(K)$ denote the absolute value, the 2-norm, and the spectral radius of any matrix K, respectively. The absolute value $|K|$ is defined as the matrix obtained by taking the absolute value of all elements of K. For any matrices $A = (a_{ij})$ and $B = (b_{ij})$, $A > B$ ($A \geq B$) means that $a_{ij} > b_{ij}$ ($a_{ij} \geq b_{ij}$) for all indices i, j.

2. Preliminaries. In this section we recall some basic definitions and known results which will be used later on.

DEFINITION 2.1. [1, 7] *Let $\mathbb{Z}^{n\times n}$ denote the set of all real $n \times n$ matrices which have all non-positive off-diagonal entries. A nonsingular matrix $A \in \mathbb{Z}^{n\times n}$ is called M-matrix if A is a nonsingular matrix and $A^{-1} \geq 0$.*

DEFINITION 2.2. [1] *For an $n \times n$ matrix $A = (a_{lj})$, we define a new matrix $\langle A\rangle = (\alpha_{lj})$, where*

$$\alpha_{lj} = \begin{cases} |a_{lj}|, & j = l \\ -|a_{lj}|, & j \neq l \end{cases},$$

and we called this matrix $\langle A\rangle$ the comparison matrix of A.

DEFINITION 2.3. [1] *If $\langle A\rangle$ is an M-matrix, then A is said to be an H-matrix.*

LEMMA 2.4. [1] *If A is a real $n \times n$ H-matrix, then $|A^{-1}| \leq \langle A\rangle^{-1}$.*

DEFINITION 2.5. [5, 7, 8] *Let $A = M - N$ be a splitting. Then this splitting is called:*

- *regular, if $M^{-1} \geq 0$ and $N \geq 0$;*
- *weak regular, if $M^{-1} \geq 0$ and $M^{-1}N \geq 0$;*
- *H-splitting if $\langle M\rangle - |N|$ is an M-matrix;*
- *H-compatible splitting if $\langle A\rangle = \langle M\rangle - |N|$.*

LEMMA 2.6. [4] *Let $A^{-1} \geq 0$ and*

$$A = M_1 - N_1 = M_2 - N_2$$

be weak regular splittings of A. In either of the following cases

a) $N_1 \leq N_2$;

b) $M_1^{-1} \geq M_2^{-1}$, $N_1 \geq 0$;

c) $M_1^{-1} \geq M_2^{-1}$, $N_2 \geq 0$;

the inequality

$$\rho(M_1^{-1}N_1) \leq \rho(M_2^{-1}N_2)$$

holds.

LEMMA 2.7. [5] *Let A be an $n \times n$ H-matrix.*

- *If the splitting $A = M - N$ is an H-splitting, then $\rho(M^{-1}N) < 1$.*
- *If the splitting $A = M - N$ is an H-compatible splitting, then it is an H-splitting.*

3. The WGS-PIA algorithm. Let $B = (D+L)+U$ be a splitting defined as in (1.6). Similar to the WPIA algorithm [15], by taking $M = \frac{1}{\omega}(D+L)$ in Eq. (1.5), we then get the following WGS-PIA algorithm

$$\mathbf{P}^{(k)} = \mathbf{P}^{(k-1)} + \omega(D+L)^{-1}(\mathbf{Q} - B\mathbf{P}^{(k-1)}), \tag{3.1}$$

where the iteration matrix is $G_{\omega gs} = I - \omega(D+L)^{-1}B$.

Obviously, when $\omega = 1$, the WGS-PIA reduces to the GS-PIA. Therefore, we can expect that the WGS-PIA has a better convergence behavior than the GS-PIA for some ω.

3.1. The convergence theorems for the J-PIA and GS-PIA. It is shown in [12, 14, 18] that collocation matrix B resulting from cubic B-spline basis functions is a totally nonnegative H-matrix. Thus, we can give more concise proofs for Theorem 1 in [14] and Theorem 2 in [18].

THEOREM 3.1. ([14, Theorem 1], [18, Theorem 2]) *Let B in* (1.4) *be the collocation matrix resulting from a cubic B-spline basis and $B = D + L + U$ be defined as in* (1.6). *Then, the iteration matrix of the J-PIA method* [14] *is $G_J = -D^{-1}(L+U)$ and the iteration matrix of the GS-PIA method* [18] *is $G_{GS} = -(D+L)^{-1}U$. Furthermore, $\rho(G_J) < 1$ and $\rho(G_{GS}) < 1$.*

Proof. From the hypothesis, we have that $B = D+L+U$ is a totally nonnegative H-matrix. Then, from Definition 2.5, we have the splitting $B = D + (L+U)$ and $B = (D+L) + U$ are H-compatible splittings, respectively. Hence, by Lemma 2.7, we obtain $\rho(G_J) < 1$ and $\rho(G_{GS}) < 1$. Thus, the proof is complete. □

In particular, we observe from the numerical tests in [18] that for cubic B-spline curve interpolations, the GS-PIA converges faster than the Jacobi-PIA. Now we give a theoretical proof.

THEOREM 3.2. *Under the hypothesis of Theorem 3.1, $\rho(G_{GS}) \le \rho(G_J)$.*

Proof. From the hypothesis, we have that $B = D + L + U$ is a totally nonnegative H-matrix, which implies $D \ge 0$, $L \ge 0$, $U \ge 0$ and $D + L$ being an H-matrix. Note that the iteration matrix of the GS-PIA is $G_{GS} = -(D+L)^{-1}U$ and the iteration matrix of the Jacobi-PIA is $G_J = -D^{-1}(L+U)$. Then we have

$$\begin{aligned}
\rho(G_{GS}) &= \rho((D+L)^{-1}U) \\
&\le \rho[|(D+L)^{-1}||U|] \\
&\le \rho[\langle D+L\rangle^{-1}U] \qquad \text{(by Lemma 2.4)} \\
&= \rho[(D-L)^{-1}U].
\end{aligned}$$

Since $D^{-1} \ge 0$, $(D-L)^{-1} \ge 0$, $D^{-1}(L+U) \ge 0$ and $(D-L)^{-1}U \ge 0$, so the splittings $\langle B\rangle = D - (L+U)$ and $\langle B\rangle = (D-L) - U$ are both weak regular. Due to $\langle B\rangle^{-1} \ge 0$ and $U \le L+U$, by Lemma 2.6, we then have

$$\rho((D-L)^{-1}U) \le \rho(D^{-1}(L+U)) = \rho(-D^{-1}(L+U)) = \rho(G_J).$$

This means $\rho(G_{GS}) \le \rho(G_J)$. The proof is complete. □

3.2. The convergence theorem. Firstly, we review Theorem 3.3 ([19, Theorems 1-2]), the convergence theorem of the extrapolation scheme of the general stationary iteration method, as follows.

LEMMA 3.3. ([19, Theorems 1-2]) *Let $S = \{\nu_j = \gamma_j + i\eta_j,\ j = 1, \cdots, n\}$ be the set of all eigenvalues of the iteration matrix G of the stationary iteration scheme*

$$x^{m+1} = Gx^m + c,\ m = 0, 1, \cdots. \tag{3.2}$$

Let γ_M, γ_m and η_M be the largest and the smallest real part, and the largest magnitude imaginary part of eigenvalues of G, respectively, and $\gamma_m \le \gamma_j \le \gamma_M < 1$ holds. Then

(1) *The extrapolation scheme of* (3.2)

$$x^{m+1} = G_\omega x^m + \omega c,\ m = 0, 1, \cdots, \tag{3.3}$$

where $G_\omega = (1-\omega)I + \omega G$, *converges if and only if* $0 < \omega < \zeta$ *with* $\zeta = \min_j\{\frac{2(1-\gamma_j)}{(1-\gamma_j)^2+\eta_j^2}\}$.

(2) *Let*

$$\omega_m = \begin{cases} \omega_1 & if \quad \vartheta \le \psi \\ \omega^* & if \quad \vartheta \ge \psi \end{cases}, \tag{3.4}$$

where $\omega_1 = \frac{1-\gamma_M}{(\gamma_M-1)^2+\gamma_j^2}$, $\omega^* = \frac{2}{2-(\gamma_m+\gamma_M)}$, $\vartheta = (1-\gamma_M)(\gamma_M-\gamma_m)$ *and* $\psi = 2\eta_M^2$, *we have*

$$\min_\omega \rho(G_\omega) \le \rho(G_{\omega_m}) = \begin{cases} \frac{\eta_M}{((\gamma_m-1)^2+\eta_M^2)^{1/2}} < 1, & if \quad \omega_m = \omega_1, \\ \frac{[(\gamma_M-\gamma_m)^2+4\beta_M^2]^{1/2}}{2-\gamma_m-\gamma_M} < 1, & if \quad \omega_m = \omega^*, \end{cases} \tag{3.5}$$

with equality holding iff (1) $\gamma_M + i\eta_M \in S$, *when* $\vartheta \le \psi$, *or* (2) $\gamma_m + i\eta_M$, $\gamma_M + i\eta_M \in S$, *when* $\vartheta \ge \psi$.

Now, we can give the convergence theorem of the WGS-PIA algorithm (3.1) for cubic B-spline curve interpolations.

THEOREM 3.4. *Let* $\mu_j = \alpha_j + i\beta_j$, $j = 1, 2, \cdots, n$, *be all eigenvalues of the matrix* $H = (D+L)^{-1}B$. *Let* $\alpha_m = \min\limits_{1\le j\le n}\{\alpha_j\}$, $\alpha_M = \max\limits_{1\le j\le n}\{\alpha_j\}$, $\beta_M = \max\limits_{1\le j\le n}|\beta_j|$, $\zeta = min_j\{\frac{2\alpha_j}{\alpha_j^2+\beta_j^2}\}$, $\vartheta = \alpha_m(\alpha_M-\alpha_m)$, $\psi = 2\beta_M^2$, $\omega_1 = \frac{\alpha_m}{\alpha_m^2+\alpha_M^2}$, $\omega^* = \frac{2}{\alpha_m+\alpha_M}$. *Then, under the hypothesis of Theorem 3.1, we have*

(1) *when* $0 < \omega < \zeta$, *the WGS-PIA (3.1) converges.*

(2) *when*

$$\omega_m = \begin{cases} \omega_1 & if \quad \vartheta \le \psi \\ \omega^* & if \quad \vartheta \ge \psi \end{cases}, \tag{3.6}$$

an upper bound of the spectral radius of the iteration matrix $G_{\omega gs}$ *of the WGS-PIA (3.1) reaches its minimum, and the following relation*

$$\min_\omega \rho(G_{\omega gs}) \le \begin{cases} \frac{\beta_M}{(\alpha_m^2+\beta_M^2)^{1/2}} < 1, & if \quad \omega_m = \omega_1, \\ \frac{[(\alpha_M-\alpha_m)^2+4\beta_M^2]^{1/2}}{\alpha_m+\alpha_M} < 1, & if \quad \omega_m = \omega^*, \end{cases}$$

holds.

Proof. We notice that $H = (D+L)^{-1}B = I + (D+L)^{-1}U = I - G_{GS}$. According to Theorem 3.1, we have that $\rho(G_{GS}) < 1$, i.e., $|\nu_j| < 1$, which implies $\gamma_m < \gamma_j < \gamma_M < 1$. Thus, by Lemma 3.3, we have derived this conclusions of (1) and (2). □

Theorem 3.4 shows that the performance of the WGS-PIA algorithm is seriously dependent on the choice of weights. We remark that the ω_m in (3.6) minimizes an upper bound of the spectral radius of the iteration matrix $G_{\omega gs}$ of WGS-PIA, and this ω_m does not minimize $\rho(G_{\omega gs})$ itself. Thus, it is a crucial and challenging task to find an approximation $\omega^\star$ of the optimal weight ω_{opt} that minimizes $\rho(G_{\omega gs})$.

4. Numerical experiments. In this section, we first present an inexpensive approach to obtain $\omega^\star$, which is a numerical approximation of ω_{opt}. This approach needs to compute the maximum and minimum eigenvalues of H in Theorem 3.4, which can be computed using the power and inverse iteration methods [17]. However, the computational complexity of these algorithms are rather large. Therefore, exploiting the

structure of collocation matrix, we first give a strategy to obtain the approximation of those eigenvalues in Section 4.1. Next, we establish the relationship between $\omega^\star$ and those eigenvalues through a large number of numerical examples in Section 4.2. Finally, we give some numerical examples to show the effectiveness of WGS-PIA in Section 4.3.

4.1. Approximation of the eigenvalues of preconditioned matrix. By Theorem 3.4, we guess that there is a correlation between $\omega^\star$ and ω_m in (3.6). The latter requires the pre-computation of the maximum and minimum eigenvalues of the preconditioned matrix H.

Fortunately, based on the experimental data so far, $\omega^\star$ changes only slightly when the eigenvalues of H are slightly perturbed. This enables us to approximately compute the maximum and minimum eigenvalues of H.

Since $D+L$ is an H-matrix and $\langle D+L\rangle = \langle D\rangle - |L|$, according to Lemma 2.7, we can derive that $\rho(D^{-1}L) = \rho(\tilde{L}) < 1$. [16, Theorem 1.11] ensures that

$$(I+\tilde{L})^{-1} = \sum_{n=0}^{\infty}(-\tilde{L})^n, \tag{4.1}$$

Thus, the following is an approximate form of H:

$$\begin{aligned} H =&(D+L)^{-1}B \\ =&I+(D+L)^{-1}U \\ =&I+(I+\tilde{L})^{-1}\tilde{U} \\ \approx&I+\tilde{U}-\tilde{L}\tilde{U} \end{aligned} \tag{4.2}$$

with $\tilde{U} = D^{-1}U$ and first two items of (4.1) are retained to yield (4.2).

It is clear that only $O(n)$ flops are needed to compute (4.2), which is much less than the $O(n^2)$ flops needed to compute H. Moreover, (4.2) is an upper bidiagonal matrix, so its eigenvalues lie on the diagonal. Thus, it is straightforward to obtain approximate maximal and minimal eigenvalues of H. In contrast, it is extremely expensive to compute the eigenvalues of lower Hessenberg matrix H directly. To sum up, the above method can approximately derive eigenvalues of H, which consumes much less CPU time than the original matrix.

4.2. Computation of $\omega^\star$. Usually, $\omega^\star$ should be in the interval $\left[\frac{\omega_m+1}{2}, \omega_m\right]$. Through numerous numerical experiments, we observe that $\omega^\star$ is linearly related to ω_m. Furthermore, by linear regression, we establish the following mathmatical model

$$\omega^\star = \frac{2\omega_m+1}{3}. \tag{4.3}$$

The following will show the relationship between the linear regression model and the sample sets. On the Archimedes spiral of Example 4.4, we select interpolation points of different sizes and pseudo-random environments as sample sets. Concretely, we use the MATLAB function `rng` to initialize the Mersenne Twister generator, selecting random number seeds from 1 to 5. We obtain a total of 100 experimental sample sets $\{(\omega_m^{(i)},\ \omega^{(i)})\}_{i=1,\cdots,100}$ from 20 groups of data points ($n = 10, 20, \ldots, 200$), each conforming to a uniform distribution over $[0, 1]$. Each $\omega_m^{(i)}$ is computed by combining (4.2) and (3.6), and the $\omega^{(i)}$ for each sample set are determined by traversing $\omega^{(i)} = 0.1 : 0.01 : 2$, where a MATLAB colon is used to denote that ω are taken every 0.01 between the interval [0.1, 2]. In Fig. 4.1, Sample points and Prediction curve denote all 100 experimental sample sets $\{(\omega_m^{(i)},\ \omega^{(i)})\}_{i=1,\cdots,100}$ and model curve (4.3), respectively. Notice that all of these sample points are distributed around the curve.

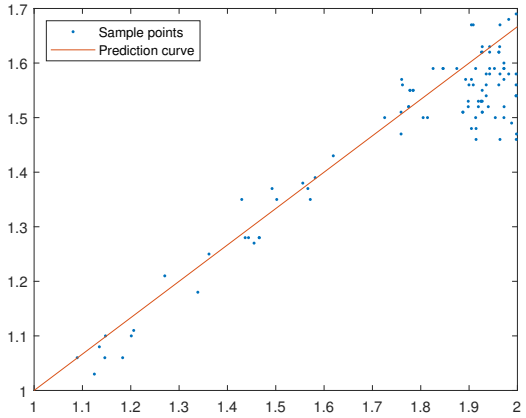


Fig. 4.1
Comparison of sample sets $\{(\omega_m^{(i)},\ \omega^{(i)})\}_{i=1,\cdots,100}$ on the Archimedes spiral of Example 4.4

To demonstrate the effectiveness of the $\omega^\star$ in (4.3), we test $\omega = 0.1 : 0.01 : 2$ and $\omega = \omega^\star$ in Examples 4.2 and 4.5 which are given in next subsection. The results are displayed in Fig.4.2, where $\epsilon^{(k)} = \max\limits_{0\le l\le n} \parallel \mathbf{c}^k(t_l) - \mathbf{q}_l \parallel$ denotes the interpolation errors at the k-th step of WGS-PIA Algorithm (3.1), and IT denotes the corresponding number of iterations. We observe that all gray lines are clustered to the right of the red line, which implys that $\omega^\star$ is a good weight for the WGS-PIA algorithm.

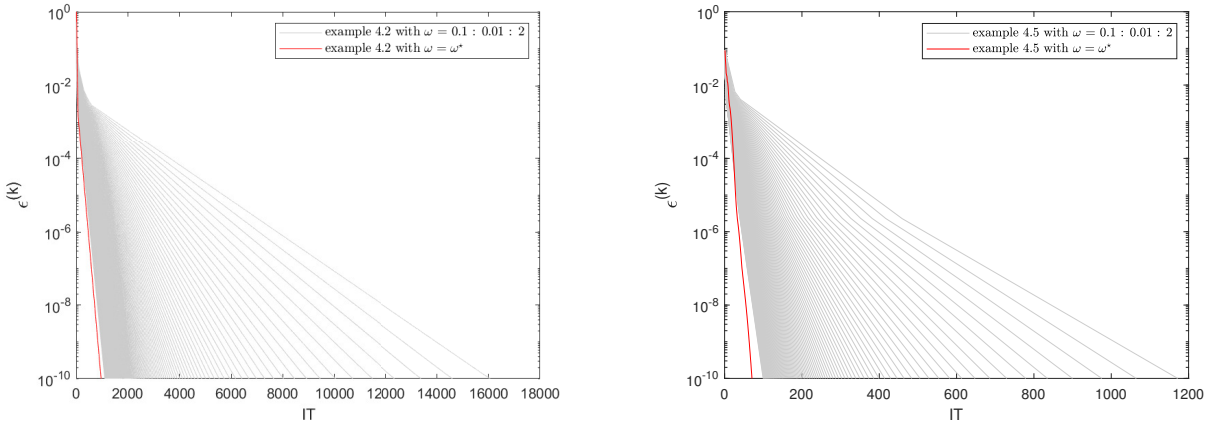


Fig. 4.2
$\epsilon^{(k)}$ for WGSPIA with different weights in Example 4.2 and 4.5

In the next subsection, we use the WGS-PIA method with $\omega^\star$ in (4.3) for computing Examples 4.1-4.5. All numerical experiments show that our approach is effective. For more details, see subsection 4.3.

4.3. Numerical examples. We use the non-uniform cubic B-spine basis to verify the effectiveness of the WGS-PIA algorithm. For comparison, we also test the GS-PIA algorithm. The performance of the algorithms is evaluated by the number of iterations (IT) and the elapsed CPU time/s (CPU/s), and CPU time is calculated by averaging 100 repeated experiments. All numerical experiments were performed on computer DESKTOP-9TC1VPG with 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz by MATLAB(R2022a).

We give five numerical examples as follows. These numerical examples are taken from [18] and

http://paulbourke.net/geometry/.

EXAMPLE 4.1. *Consider a parametric function given by*

$$\begin{cases} x = \cos u \left(2 - \cos\left(\frac{2u}{2k+1}\right)\right) \\ y = \sin u \left(2 - \cos\left(\frac{2u}{2k+1}\right)\right) \\ z = -\sin\left(\frac{2u}{2k+1}\right) \end{cases}$$

where $0 \leq u \leq (4k+2)\pi$. *It is called Cinquefoil Knot if* $k = 2$. *The 500 interpolation points on it were selected non-uniformly.*

EXAMPLE 4.2. *Consider the helix given by*

$$\begin{cases} x = rcos(\tau(\theta_2 - \theta_1) + \theta_1) \\ y = rsin(\tau(\theta_2 - \theta_1) + \theta_1) \\ z = h\tau \end{cases}$$

where $r = 30, h = 50, \theta_1 = \frac{\pi}{6}, \theta_2 = 40\pi$. *The value of* τ *at 2000 in [0, 1] is uniformly selected to obtain the interpolation points.*

EXAMPLE 4.3. *Consider 400 interpolation points on the Freeths Nephroid* $\rho = 1 + 2sin(\theta/2)(0 \leq \theta \leq 6\pi)$.

EXAMPLE 4.4. *Consider the data interpolation of 3000 points sampled from Archimedes spiral* $\rho = \theta(0 \leq \theta \leq 30\pi)$.

EXAMPLE 4.5. *Given 987 scattered interpolation points.*

We select Examples 4.1-4.2 for the 3D curve fits, Examples 4.3-4.5 for the 2D curve fits. Among them, interpolation points of the Examples 4.1 and 4.3 are selected at a smaller scale than Example 4.2 and Example 4.4. Example 4.5 was given as an example of a set of scatter points.

In Table 4.1-4.5, we record the number of iterations (the elapsed CPU time) required by GS-PIA algorithm and WGS-PIA algorithm when $\epsilon^{(k)} < 10^{-6}, 10^{-7}, \cdots, 10^{-11}$. The weight $\omega^\star$ are computed via (4.3), and its values(CPU time) are listed independently in the second column of the tables.

Table 4.1
The number of iteration steps (CPU time) for some given tolerances in Example 4.1

Algorithm	$\omega^\star$ (CPU/s)	IT(CPU/s)					
		$\epsilon^{(k)} < 10^{-6}$	$\epsilon^{(k)} < 10^{-7}$	$\epsilon^{(k)} < 10^{-8}$	$\epsilon^{(k)} < 10^{-9}$	$\epsilon^{(k)} < 10^{-10}$	$\epsilon^{(k)} < 10^{-11}$
GS-PIA	-	322(0.0296)	432(0.0387)	542(0.0492)	652(0.0594)	762(0.0679)	872(0.0796)
WGS-PIA	1.6396(0.0015)	195(0.0212)	262(0.0293)	328(0.0340)	395(0.0400)	462(0.0476)	528(0.0541)

Table 4.2
The number of iteration steps (CPU time) for some given tolerances in Example 4.2

Algorithm	$\omega^\star$(CPU/s)	IT(CPU/s)					
		$\epsilon^{(k)} < 10^{-6}$	$\epsilon^{(k)} < 10^{-7}$	$\epsilon^{(k)} < 10^{-8}$	$\epsilon^{(k)} < 10^{-9}$	$\epsilon^{(k)} < 10^{-10}$	$\epsilon^{(k)} < 10^{-11}$
GS-PIA	-	774(1.6800)	980(2.0701)	1186(2.7445)	1391(3.3275)	1600(3.5826)	1825(3.6609)
WGS-PIA	1.4380(0.0345)	467(1.0994)	592(1.4877)	715(1.7682)	839(1.9915)	965(2.2573)	1101(2.6753)

Table 4.3
The number of iteration steps (CPU time) for some given tolerances in Example 4.3

Algorithm	$\omega^\star$ (CPU/s)	IT(CPU/s)					
		$\epsilon^{(k)} < 10^{-6}$	$\epsilon^{(k)} < 10^{-7}$	$\epsilon^{(k)} < 10^{-8}$	$\epsilon^{(k)} < 10^{-9}$	$\epsilon^{(k)} < 10^{-10}$	$\epsilon^{(k)} < 10^{-11}$
GS-PIA	-	1113(0.0870)	1575(0.1237)	2038(0.1621)	2500(0.2039)	2962(0.2390)	3424(0.2750)
WGS-PIA	1.6601(0.0012)	670(0.0469)	948(0.0738)	1226(0.1137)	1504(0.1353)	1782(0.1589)	2060(0.1858)

Table 4.4
The number of iteration steps (CPU time) for some given tolerances in Example 4.4

Algorithm	$\omega^\star$ (CPU/s)	IT(CPU/s)					
		$\epsilon^{(k)} < 10^{-6}$	$\epsilon^{(k)} < 10^{-7}$	$\epsilon^{(k)} < 10^{-8}$	$\epsilon^{(k)} < 10^{-9}$	$\epsilon^{(k)} < 10^{-10}$	$\epsilon^{(k)} < 10^{-11}$
GS-PIA	-	1903(5.9570)	2540(7.9414)	3176(9.8952)	3812(11.8881)	4449(13.8767)	5085(15.8941)
WGS-PIA	1.6740(0.0648)	1144(4.2880)	1527(5.7635)	1909(7.0314)	2292(7.9557)	2674(9.2660)	3056(10.5765)

Table 4.5
The number of iteration steps (CPU time) for some given tolerances in Example 4.5

Algorithm	$\omega^\star$ (CPU/s)	IT(CPU/s)					
		$\epsilon^{(k)} < 10^{-6}$	$\epsilon^{(k)} < 10^{-7}$	$\epsilon^{(k)} < 10^{-8}$	$\epsilon^{(k)} < 10^{-9}$	$\epsilon^{(k)} < 10^{-10}$	$\epsilon^{(k)} < 10^{-11}$
GS-PIA	-	50(0.0282)	65(0.0340)	80(0.0402)	96(0.0474)	111(0.0549)	126(0.0647)
WGS-PIA	1.5094(0.0076)	37(0.0208)	45(0.0269)	56(0.0275)	64(0.0354)	71(0.0389)	81(0.0443)

In Figs. 4.3-4.7, we display the initial control node and their WGS-PIA interpolation figures for the 1-st iteraion and the 10-th iteraions, as well as the relation diagrams of iteration steps and interpolation errors of the GS-PIA and WGS-PIA.

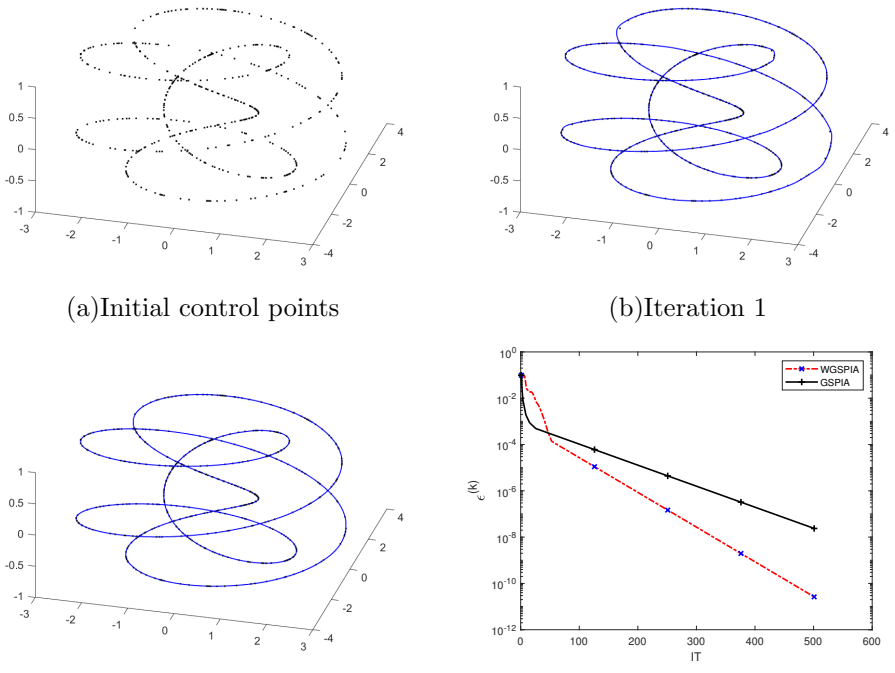
It can be seen with Tables 4.1-4.5 that, at least from the interpolation error less than 10^{-6}, the number of iterations of WGS-PIA is only about 60% of that of GS-PIA. This conclusion can be intuitively perceived by the graphs in the lower right corner of Figs. 4.3-4.7. Initially, the interpolation errors of WGS-PIA are reduced more slowly compared to GS-PIA. When ϵ is roughly less than 10^{-4}, the errors of WGS-PIA decay faster than that of GS-PIA.

On the other hand, WGS-PIA takes less CPU time compared to GS-PIA with the same interpolation error. And in the WGS-PIA algorithm, the computation time of $\omega^\star$ is almost negligible compared to the iteration time. Since the weight $\omega^\star$ only needs to be solved once in each experiment, as the required convergence accuracy increases, the percentage of computation time for this part becomes increasingly smaller.

5. Conclusions. In this paper, we have developed the WGS-PIA algorithm by combining the GS-PIA [18] with the WPIA [15]. In Theorem 3.4, we have shown that there always exists a positive weight ω such that WGS-PIA algorithm converges for cubic B-spline curve interpolations. Moreover, we have given an ω_m that minimizes an upper bound of the spectral radius of the iteration matrix $G_{\omega gs}$ of the WGS-PIA (3.1).

Meanwhile, we have given a more concisely proof in Theorem 3.1, which shows the convergence of the GS-PIA [18] and the J-PIA [14] by splitting theories [5]. In the numerical experiments of [18], we observed GS-PIA has a better performance than J-PIA for cubic B-spline curve interpolations. We then proved it theoretically, and this conclusion has been shown in Theorem 3.2.

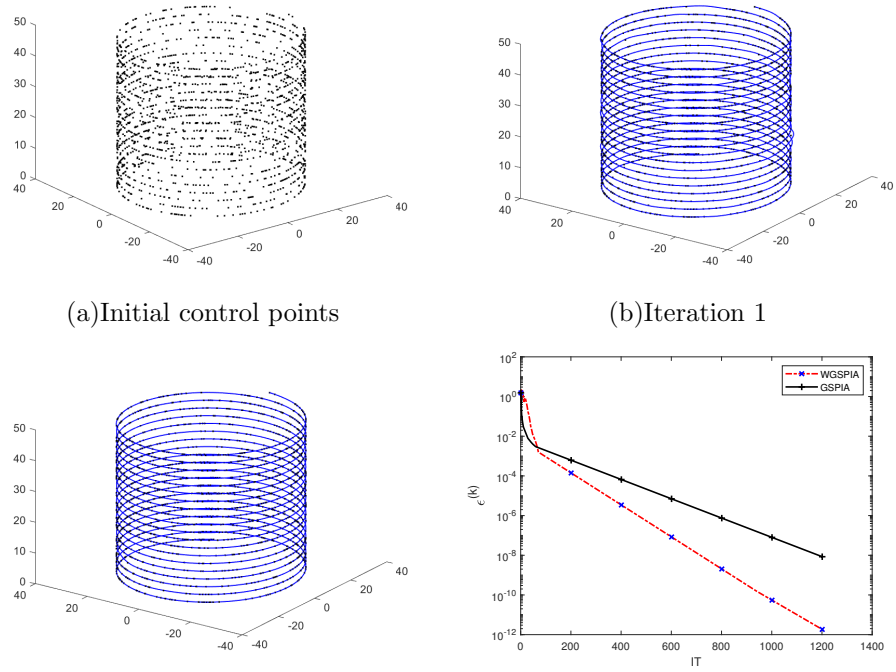
Theoretically, we can chose ω to be any constant in the interval $(0,\ \zeta)$ with ζ defined in Theorem 3.4.



(a)Initial control points (b)Iteration 1

(c)Iteration 10 (d)IT vs $\epsilon^{(k)}$

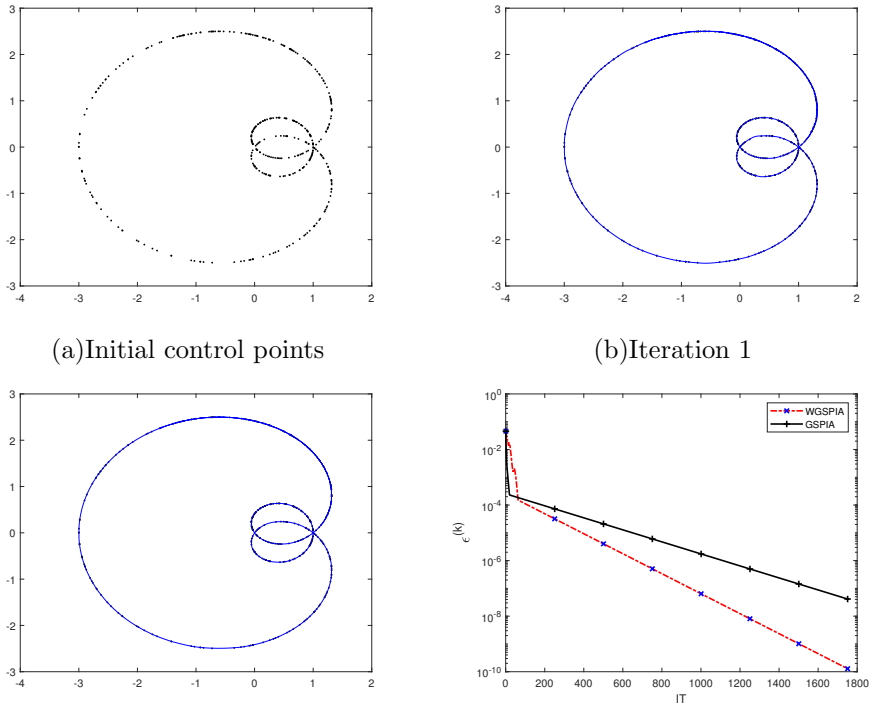
Fig. 4.3
The initial control points, the fitting curves, and the iteration number versus relative error for Example 4.1



(a)Initial control points (b)Iteration 1

(c)Iteration 10 (d)IT vs $\epsilon^{(k)}$

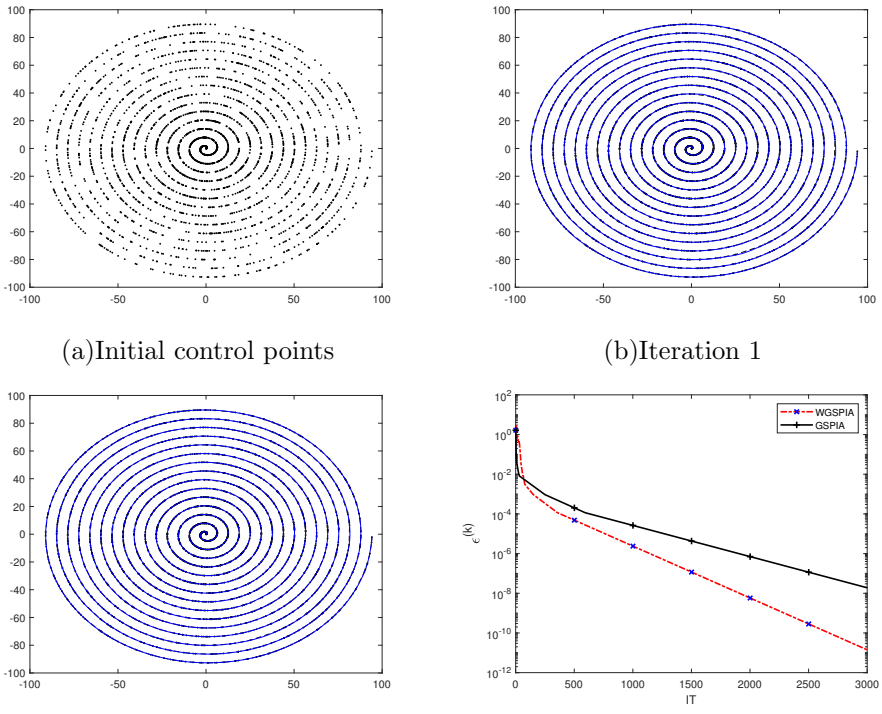
Fig. 4.4
The initial control points, the fitting curves, and the iteration number versus relative error for Example 4.2.



(a)Initial control points (b)Iteration 1

(c)Iteration 10 (d)IT vs $\epsilon^{(k)}$

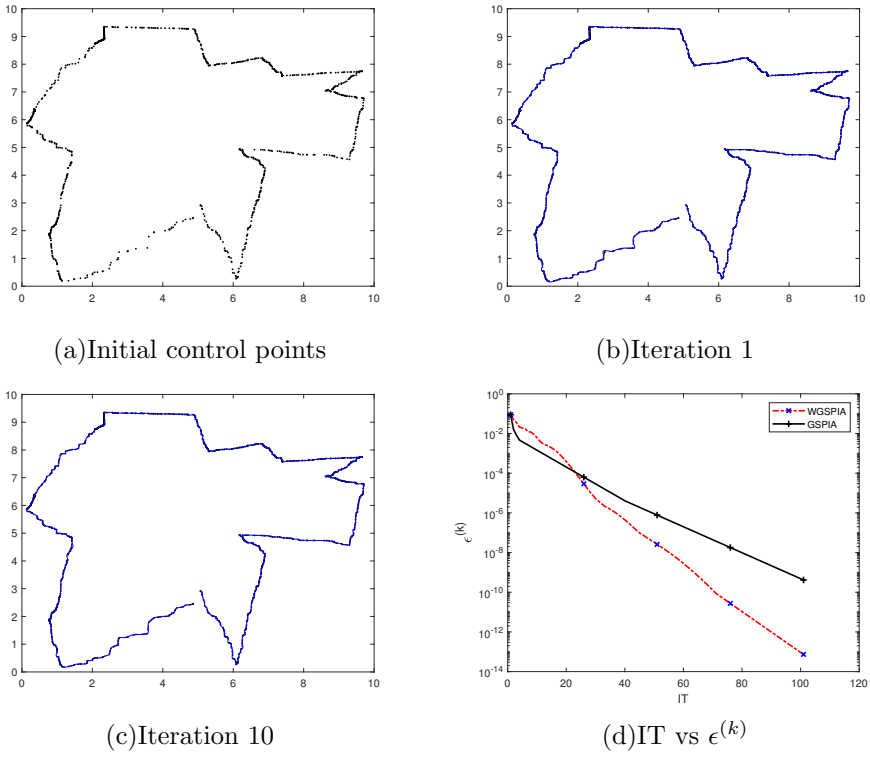
Fig. 4.5
The initial control points, the fitting curves, and the iteration number versus relative error for Example 4.3.



(a)Initial control points (b)Iteration 1

(c)Iteration 10 (d)IT vs $\epsilon^{(k)}$

Fig. 4.6
The initial control points, the fitting curves, and the iteration number versus relative error for Example 4.4.



(a)Initial control points (b)Iteration 1

(c)Iteration 10 (d)IT vs $\epsilon^{(k)}$

Fig. 4.7
The initial control points, the fitting curves, and the iteration number versus relative error for Example 4.5.

In particular, WGS-PIA reduces to GS-PIA when $\omega = 1$ in (3.1). However, setting ω to be a value different from 1 does not always result in improved performance compared to the GS-PIA algorithm. Over a large set of random test cases, we have given formula (4.3) to compute a good numerical approximation $\omega^\star$ of the optimal weight ω_{opt} inexpensively, such that WGS-PIA has fewer iterations and less CPU time than GS-PIA. Theoretically, ω_{opt} is strongly dependents on the structure and properties of the collocation matrix, WGS-PIA under other bases therefore deserves further study in depth.

6. Acknowledgement. This work was supported by the Postgraduate Scientific Research Innovation Project of Hunan Province (CX20220953), Research Foundation of Education Bureau of Hunan Province under No. 22C0147 and FCT (Fundação para a Ciência e a Tecnologia) within the Projects UIDB/00013/2020 and UIDP/00013/2020.

7. Declarations. Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

REFERENCES

[1] F. Andreas and B. S. Daniel, *H-Splittings and two-stage iterative methods,* Numer. Math., 63(1992), 345-356.
[2] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.
[3] J. M. Carnicer, J. Delgado and J. M. Peña, *Richardson method and totally nonnegative linear systems,* Linear Algebra Appl., 433(2010), 2010-2017.
[4] L. Elsner, *Comparisons of weak regular splittings and multisplitting methods,* Numer. Math., 56(1989), 283-289.
[5] A. Frommer and D. B. Szyld, *H-splitting and two-stage iterative methods*, Numer. Math. 63 (1992) 345-356.
[6] A. Hadjidimos and A. Yeyios, *The principle of extrapolation in connection with the accelerated overrelaxation method,* Linear Algebra Appl., 30(1980).

[7] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
[8] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1991.
[9] H. W. Lin, *Geometric iterative method and its application,* J. Comput. Aided Des. Comput. Graph., 4(2015), 582-589.
[10] H. W. Lin, H. J. Bao and G. J. Wang, *Totally positive bases and progressive iteration approximation,* Comput. Math. Appl., 50(2015), 575-586.
[11] H. W. Lin, T. Maekawa and C. Y. Deng, *Survey on geometric iterative methods and their application,* Comput. Aided Des., 95(2018), 40-51.
[12] C. Z. Liu, X. L. Han and J. C. Li, *Progressive Iterative Approximation by Extension of Cubic Uniform B-spline Curves,* J. Comput. Aided Des. Comput. Graph., 31(2019), 899-910.
[13] C. Z. Liu and Z. Y. Liu, *Progressive Iterative Approximation with Preconditioners,* Math., 8(2020), 1503.
[14] X. Y. Liu and C. Y. Deng, *Jacobi-PIA algorithm for non-uniform cubic B-spline curve interpolation,* J. Comput. Aided Des. Comput. Graph., 27(2015), 485-491.
[15] L. Z. Lu, *Weighted progressive iteration approximation and convergence analysis,* Comput. Aided Geom. Des., 27(2010), 129-137.
[16] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd edition, SIAM, Philadelphia, PA, USA, 2003.
[17] Demmel, James W, Applied numerical linear algebra, SIAM, Society for Industrial and Applied Mathematics, 1997.
[18] Z. H. Wang, Y. J. Li and C. Y. Deng *Convergence proof of GS-PIA algorithm,* J. Comput. Aided Des. Comput. Graph., 30(2018), 60-66.
[19] A. Yeyios, *On the optimization of an extrapolation method,* Linear Algebra Appl., 57 (1984), 191-203.
[20] L. Zhang, J. Q. Tan, X. Y. Ge and G. Zheng, *Generalized B-splines' geometric iterative fitting method*, J. Comput. Appl. Math., 329 (2018), 331-343.
[21] F.Z. Shi, CAGD & NURBS, M. Beijing: Higher Education Press, 2001: 211-248(in Chinese).