Progressive Iterative Approximation with Preconditioners

Chengzhi Liu 1,2 and Zhongyun Liu 3,*

1 School of Mathematics and Statistics, Central South University, Changsha 410083, China; 162101002@csu.edu.cn
2 School of Mathematics and Finance, Hunan University of Humanities, Science and Technology, Loudi 417000, China
3 School of Mathematics and Statistics, Changsha University of Science and Technology, Changsha 410076, China
* Correspondence: liuzhongyun@263.net

Received: 5 August 2020; Accepted: 1 September 2020; Published: 4 September 2020

Abstract: The progressive iterative approximation (PIA) plays an important role in curve and surface fitting. By using the diagonally compensated reduction of the collocation matrix, we propose the preconditioned progressive iterative approximation (PPIA) to improve the convergence rate of PIA. For most of the normalized totally positive bases, we show that the presented PPIA can accelerate the convergence rate significantly in comparison with the weighted progressive iteration approximation (WPIA) and the progressive iterative approximation with different weights (DWPIA). Furthermore, we propose an inexact variant of the PPIA (IPPIA) to reduce the computational complexity of the PPIA. We introduce the inexact solver of the preconditioning system by employing some state-of-the-art iterative methods. Numerical results show that both the PPIA and the IPPIA converge faster than the WPIA and DWPIA, while the elapsed CPU times of the PPIA and IPPIA are less than those of the WPIA and DWPIA.

Keywords: PIA; WPIA; diagonally compensated reduction; preconditioner; inexact solver; convergence

1. Introduction

The progressive iterative approximation (PIA) is an important iterative method for fitting a given set of data points. Due to its clear geometric meaning, stable convergence and simple iterative scheme, the PIA has intrigued researchers for decades. In particular, said methodology has achieved great success in geometric design [1,2], data fitting [3,4], reverse engineering [5] and NURBS solid generation [6–8]. For more detailed research on this topic, we refer the reader to read a recent survey [9]. However, the PIA converges very slowly, especially when the number of interpolating points increases to some extent. This is because the collocation matrices resulting from some normalized totally positive (NTP) bases are usually ill-conditioned; see [10,11]. Therefore, much more attention was paid to accelerating the convergence rate of PIA; see the weighted progressive iteration approximation (WPIA) (also named modified Richardson method) in [12,13]; the QR-PIA in [11]; the Jacobi-PIA in [14]; the Chebyshev semi-iterative method in [15]; the progressive iterative approximation with different weights (DWPIA) in [16]; the GS-PIA in [17]; the PIA with mutually different weights in [18]; the Schulz iterative method in [19]; and a lot literature therein.

There are other remedy strategies, such as locally interpolating, i.e., interpolating partial points selected from the total given points, which results in the local progressive-iterative approximation format [20], the extended progressive-iterative approximation format [21] and the progressive iterative approximation format for least square fitting [22].
Let \( \{ b_j(t) \}_{j=0}^{n} \) be an NTP basis, such as the Bernstein basis \([23]\) or the Said–Ball basis \([24]\). Given a set of data points \( \{ v_i \}_{i=0}^{n} \) in \( \mathbb{R}^2 \) or \( \mathbb{R}^3 \), we assign a parameter value \( t_i \) to the \( i \)-th point \( v_i \). For PIA, the initial interpolating curve is given by

\[
\gamma^{(0)}(t) = \sum_{i=0}^{n} u_i^{(0)} b_i(t), \quad u_i^{(0)} = v_i, \quad i = 0, 1, \ldots, n,
\]

and the \((k + 1)\)-th interpolating curve can be generated by

\[
\gamma^{(k+1)}(t) = \sum_{i=0}^{n} u_i^{(k+1)} b_i(t), \quad k = 0, 1, 2, \ldots,
\]

where

\[
u_i^{(k+1)} = u_i^{(k)} + \delta_i^{(k+1)}, \quad \delta_i^{(k+1)} = v_i - \gamma^{(k)}(t_i).
\]

Equation (1) can be rewritten as the following form

\[
u_i^{(k+1)} = \nu_i^{(k)} = \nu_i - \sum_{j=0}^{n} u_j^{(k)} b_j(t_i), \quad i = 0, 1, \ldots, n,
\]

which generates a sequence of control polygons \( u_i^{(k+1)} \), for \( k = 0, 1, 2, \ldots \).

Let \( \Delta^{(k)} = \left[ \delta_0^{(k)}, \delta_1^{(k)}, \ldots, \delta_n^{(k)} \right]^T \), \( U^{(k)} = \left[ u_0^{(k)}, u_1^{(k)}, \ldots, u_n^{(k)} \right]^T \), \( V = [v_0, v_1, \ldots, v_n]^T \). Then the matrix form of Equations (1) and (2) is

\[
U^{(k+1)} = U^{(k)} + \Delta^{(k)} = (I - B)U^{(k)} + V,
\]

where \( U^{(0)} = V \), \( \Delta^{(k)} = V - BU^{(k)} \), \( I \) is the identity matrix and \( B \) is the collocation matrix resulting from the basis \( \{ b_0(t), b_1(t), \ldots, b_n(t) \} \) at \( t_i, i = 0, 1, \ldots, n \). We note in [23] that the collocation matrix \( B \) resulting from any NTP basis is a stochastic totally positive matrix.

If \( \lim_{k \to \infty} \gamma^{(k)}(t_i) = v_i \), then the sequence of curves \( \gamma^{(k)}(t) \) is said to have the PIA property and Equation (2) is referred to as the PIA format. It is shown in [25] that the PIA property holds for any NTP basis.

In fact, the PIA is mathematically equivalent to the Richardson iteration for solving the collocation matrix equation

\[
BU = V;
\]

see [15,26]. Therefore, the state-of-the-art iterative method, the generalized minimal residual (GMRES) method for solving general linear systems of equations, was suggested in [26] for solving (4). It is known that the GMRES method works efficiently when the coefficient matrix \( B \) is positive definite. It is true that the GMRES has better convergence behavior than PIA in the same required precision when fitting 19 points on the helix. However, we found that the collocation matrix \( B \) resulting from some NTP bases is not always positive definite, even if \( B \) is a totally positive matrix. This led us to further study the PIA. Obviously, the convergence of PIA depends on the spectral radius of the iteration matrix \( I - B \). The smaller the spectral radius is, the faster the PIA converges. This motivated us to use the preconditioning techniques to reduce the spectral radius of the iteration matrix \( I - B \). The PIA consists of the interpolatory PIA and the approximating PIA [9]. In the first case, the number of the control points is the same as that of the given data points. In the second case, the number of the selected control points is less than that of the given data points. In this paper, we focus on preconditioning the interpolatory PIA. The proposed preconditioning technique can be extended straightforwardly to the approximating PIA.

The paper is organized as follows. In Section 2, we first exploit the diagonally compensated reduction of the collocation matrix \( B \) to construct a class of preconditioners for \( B \); then we establish the...
PPIA (preconditioned PIA) format and analyze its convergence. In order to improve the computational efficiency of the PPIA, we further propose an IPPIA (inexact PPIA) format and analyze its convergence in Section 3. Finally, some numerical examples are given to illustrate the effectiveness of our proposed methods in Section 4.

2. The Preconditioned Progressive Iterative Approximation (PPIA)

As mentioned earlier, the PIA does always converge although slowly. In this section, we therefore consider the preconditioning technique to accelerate the convergence rate of PIA. Preconditioning means replacing the initial system (4) with the system

\[ P^{-1}B\mathbf{u} = P^{-1}\mathbf{v}, \]

where \( P \) is an approximation of \( B \) with the following properties: \( P^{-1}B \) is well conditioned or has few extreme eigenvalues, and \( P\mathbf{z} = \mathbf{d} \) is easy to solve or \( P^{-1} \) is easy to obtain.

A careful and problem-dependent choice of \( P \) can often make the condition number of \( P^{-1}B \) much smaller than that of \( B \) and thus accelerate convergence dramatically.

2.1. Preconditioners

In this subsection, we focus on preconditioner \( P \) for the collocation matrix \( B \) resulting from an NTP basis. Very often, the preconditioner \( P \) should be chosen to preserve certain structures of the original matrix \( B \). There are many applications that generate structured matrices, and by exploiting the structure, one may be able to design faster and/or more accurate algorithms; see for example [27–29]; furthermore, structure may also help in producing solutions which have more precise physical meanings. Structure comes in many forms, including Hamiltonian, Toeplitz, Vandermonde matrices and so on. Note that \( B \) is a stochastic matrix, so we hope that the preconditioner \( P \) is also a stochastic matrix. Moreover, in order to get \( P^{-1} \) easily, we restrict \( P \) to be a banded matrix with small bandwidth. Based on those two requirements, we construct \( P \) as follows.

We first split \( B \) into \( B = B_q + R \), where

\[
R = \begin{pmatrix}
0 & \cdots & 0 & b_{q+1}(t_0) & \cdots & b_n(t_0) \\
\vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & b_n(t_{q+1}) & \vdots \\
b_0(t_{q+1}) & \ddots & \ddots & \ddots & 0 & \vdots \\
\vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\
b_0(t_n) & \cdots & b_{q+1}(t_n) & 0 & \cdots & 0
\end{pmatrix} \geq 0, \tag{6}
\]

and \( B_q = B - R \) is a banded matrix with bandwidth \( 2q + 1(0 \leq q \leq n) \). Then we define a diagonal matrix

\[
D = \text{diag}(d_0, d_1, \cdots, d_n) \tag{7}
\]

such that \( De = Re \) with \( e = (1, 1, \cdots, 1)^T \). Finally, let

\[
P = B_q + D, \tag{8}
\]

be the diagonally compensated reduction matrix of \( B \). The matrix \( R \) is called the reduced matrix and \( D \) is called the compensation matrix for \( R \); see for instance [30,31]. It is easy to check that \( P \) is a stochastic banded matrix and its construction time is small enough and can be neglected, because it only involves a matrix-vector multiplication. If \( P \) in (8) is invertible, it can serve as a preconditioner for the system (4). Unfortunately, we cannot give a theoretical proof about the nonsingularity of \( P \),
but numerous numerical experiments demonstrate \( P \) is invertible for different order \( n \) with various \( q \). Therefore, we have
\[
P^{-1}B\mathbf{u} = P^{-1}V.
\]
(9)

Accordingly, the PIA for (9) becomes the following format
\[
U^{(k+1)} = (I - P^{-1}B)U^{(k)} + P^{-1}V, \quad k = 0, 1, \ldots,
\]
where \( U^{(0)} = V \), or equivalently
\[
U^{(k+1)} = U^{(k)} + P^{-1}\Delta^{(k)},
\]
where \( \Delta^{(k)} = V - BU^{(k)} \).

For convenience, we refer to (10) as the PPIA format.

2.2. Convergence Analysis of PPIA

Before analyzing the convergence, we first investigate some properties of the collocation matrix \( B \) resulting from the Bernstein basis.

**Proposition 1.** For the Bernstein basis, if \( t_i = \frac{i}{n}, i = 0, 1, \ldots, n \), then the entries of the collocation matrix \( B \) are
\[
b_j(t_i) = \binom{n}{j}(1 - t_i)^{n-j}t_i^j, \quad i = 0, 1, \ldots, n; \quad j = 0, 1, \ldots, n,
\]
and possess the following properties:

(a) The diagonal entry \( b_j(t_j) \) is the maximum of all entries which locate at the \( j \)-th row and \( j \)-th column; i.e.,
\[
b_j(t_j) = \max_{0 \leq i \leq n} b_j(t_i) \quad \text{and} \quad b_j(t_i) = \max_{0 \leq i \leq n} b_i(t_j).
\]

(b) \( b_j(t_j) > b_j(t_{j-1}) > \cdots > b_j(t_0) \geq 0 \), and \( b_j(t_j) > b_j(t_{j+1}) > \cdots > b_j(t_n) \geq 0 \).

(c) \( b_j(t_j) > b_{j-1}(t_j) > \cdots > b_0(t_j) \geq 0 \), and \( b_j(t_j) > b_{j+1}(t_j) > \cdots > b_n(t_j) \geq 0 \).

**Proof.** It was proven in [23] that the Bernstein basis has the unimodality property in the interval \([0,1]\); i.e., \( b_j(t) \) has a unique extremum at \( t = j/n \) in the interval \([0,1]\). Moreover, for any fixed \( t^* \in [0,1] \), there exists a corresponding index \( j \) such that
\[
b_0(t^*) \leq \cdots \leq b_{j-1}(t^*) \leq b_j(t^*) \quad \text{and} \quad b_j(t^*) \geq b_{j+1}(t^*) \geq \cdots \geq b_n(t^*).
\]

In other words, \( b_j(t) \) reaches its maximum at the value \( t^* \) which is the closest to \( j/n \); thus Property (a) holds. Properties (b) and (c) can be proved in a similar way, by using the unimodality property of the Bernstein basis.

On one hand, according to the definition of \( D \) in (7), we have
\[
\|D\|_\infty = \|R\|_\infty.
\]
(11)

From (8) and (11), we further have
\[
\|P - B\|_\infty = \|D - R\|_\infty \leq 2\|R\|_\infty.
\]
(12)

On the other hand, from (6), we have
\[
\|R\|_\infty = \max_{0 \leq j \leq n} \sum_{|i-j| > q} b_i(t_j)
\]
\[
= \max_{0 \leq j \leq n} \left\{ b_0(t_j) + \cdots + b_{j-q-1}(t_j) + b_{j+q+1}(t_j) + \cdots + b_n(t_j) \right\},
\]
where the term \(b_k(t_j) = 0\), if \(|j-k| \leq q\). By Proposition 1, we have

\[
b_{j+q+1}(t_j) > \cdots > b_n(t_j) = 0, \ b_{j-q-1}(t_j) > \cdots > b_0(t_j) = 0.
\]

For each fixed \(j\), it is easy to verify that both \(b_{j+q+1}(t_j)\) and \(b_{j-q-1}(t_j)\) are monotonically decreasing and finally decay to zero, as \(q\) increases. Hence, any of sum of each row’s entries outside of the \(q\)-th diagonal decays to 0. This means that \(\|R\|_\infty\) approaches zero as \(q \to n\). Combined with (12), we conclude that the matrix \(P\) is a good approximation of \(B\). Thus, the spectrum of \(P^{-1}B\) approaches 1 for large \(q\).

**Remark 1.** We remark here that the spectral radius \(\rho(I - P^{-1}B)\) may decrease as \(q\) increases; i.e., the bigger the \(q\) is, the faster the PPIA format converges. However, according to (10), one needs to compute \(P^{-1}\Delta^{(k)}\) or equivalently solve an additional matrix equation \(PX^{(k)} = \Delta^{(k)}\) at the \(k\)th iteration. This will result in a large computational complexity, especially for large \(q\). Therefore, we need to seek a tradeoff between the convergence rate and computational complexity. Experimentally, we found that \(q \approx \frac{2}{3}\) is a suitable choice.

**Remark 2.** We emphasize that most of the NTP bases have the unimodality property, so Proposition 1 also holds for other NTP bases with unimodality. This means that our PPIA will work well for such a class of bases, such as the Said–Ball basis. This was experimentally verified in our numerical tests.

3. The Inexact Preconditioned Progressive Iterative Approximation (IPPIA)

As mentioned in Remark 1, we need to compute \(P^{-1}\Delta^{(k)}\) or equivalently solve \(PX^{(k)} = \Delta^{(k)}\) at the \(k\)-th iteration. This is very costly and impractical in actual implementations for large \(n\). To reduce the computational complexity, we propose an IPPIA format in which we inexactly solve \(PX^{(k)} = \Delta^{(k)}\) by employing a state-of-the-art iterative method, that is, the conjugate gradient (CG) method for solving the related system of normal equations.

3.1. The Inexact Solver

Denote by \(\{\hat{U}^{(k)}\}_{k=0}^\infty\) the iteration sequence generated from IPPIA format defined below. Since the sequence \(\{\hat{U}^{(k)}\}_{k=0}^\infty\) is an approximation of \(\{U^{(k)}\}_{k=0}^\infty\), \(\Delta^{(k)}\) in \(PX^{(k)} = \Delta^{(k)}\) should be replaced by \(\hat{\Delta}^{(k)} = V - BU^{(k)}\).

Now, let \(X^{(l,k)}\) be the \(l\)-th approximation of the solution of

\[
PX^{(k)} = \hat{\Delta}^{(k)},
\]

which can be obtained by employing the CG method to solve the system of normal equations

\[
p^Tpx^{(k)} = p^T\hat{\Delta}^{(k)}.
\]

The CG method used here can be thought of as an inner iterative format and terminates if

\[
\|p^TPX^{(l,k)}\|_F / \|p^T\hat{\Delta}^{(k)}\|_F < \tau_k,
\]

where \(\tau_k\) is the prescribed stopping tolerance and \(\|\cdot\|\) is the Frobenius norm.

Once the approximation \(X^{(l,k)}\) to (13) satisfies (15), we can form \(\hat{U}^{(k+1)}\) by the following outer iteration

\[
\hat{U}^{(k+1)} = \hat{U}^{(k)} + X^{(l,k)}.
\]

We refer to (16) as the IPPIA format.
Theorem 1. Let $P$ be the preconditioner defined as in (8) for $B$ and $U^*$ be the exact solution to (4). Suppose that the CG method starts from an initial guess $X^{(0,k)}$, terminates if (15) is satisfied and produces an inner iterative sequence $\{X^{(l,k)}\}_{k=0}^{\infty}$ for each outer iterative index $k$, where $X^{(l,k)}$ is an approximation of $P^{-1}\Delta^{(k)}$. Then, for the iterative sequence $\{\hat{U}^{(k)}\}_{k=0}^{\infty}$ generated by IPPIA, we have

$$
\left\|\hat{U}^{(k+1)} - U^*\right\|_F \leq \tau_k \left\|\hat{U}^{(k)} - U^*\right\|_F,
$$

where $\tau_k = \|I - P^{-1}B\|_F + \tau_k\|P^{-1}\|_F P^{-1}\|\|B\|_F$. In particular, if

$$
\tau_{\text{max}} = \|I - P^{-1}B\|_F + \tau_{\text{max}}\|P^{-1}\|_F P^{-1}\|\|B\|_F < 1,
$$

then the iterative sequence $\{\hat{U}^{(k)}\}_{k=0}^{\infty}$ converges to $U^*$, where $\tau_{\text{max}} = \max_k \{\tau_k\}$.

Proof. Note that $\Delta^{(k)} = V - BU^{(k)}$, $\hat{\Delta}^{(k)} = V - B\hat{U}^{(k)}$, from (10), we have

$$
\begin{align*}
U^{(k+1)} &= U^{(k)} + P^{-1}\left(V - BU^{(k)}\right) \\
&= U^{(k)} + P^{-1}\left(V - B\hat{U}^{(k)} + B\hat{U}^{(k)} - BU^{(k)}\right) \\
&= U^{(k)} + P^{-1}\hat{\Delta}^{(k)} + P^{-1}B\left(\hat{U}^{(k)} - U^{(k)}\right). \\
\end{align*}
$$

Let $U^*$ be the exact solution of (4). Then

$$
U^{(k+1)} - U^* = \left(I - P^{-1}B\right)\left(U^{(k)} - U^*\right). \tag{21}
$$

According to (16), (20) and (21), we have

$$
\begin{align*}
p^T\left(P - I\right)\left(U^{(k+1)} - U^*\right) &= p^T\left(U^{(k+1)} - U^{(k+1)} + U^{(k+1)} - U^*\right) \\
&= p^T\left(\hat{U}^{(k+1)} + X^{(l,k)}\right) - p^T\left(U^{(k)} + P^{-1}\hat{\Delta}^{(k)} + P^{-1}B\left(U^{(k)} - U^{(k)}\right)\right) \\
&= p^T\left(I - P^{-1}B\right)\left(U^{(k)} - U^*\right) + p^T\left(p - P\right)X^{(l,k)} - p^T\hat{\Delta}^{(k)}. \tag{22}
\end{align*}
$$
Due to the nonsingularity of $P$, (22) can be transformed into
\[ \hat{U}^{(k+1)} - U^* = \left( I - P^{-1}B \right) \left( \hat{U}^{(k)} - U^* \right) + p^{-1}p^{-T} \left( p^T \hat{p}X^{(k)} - p^T \hat{p} \Delta^{(k)} \right). \]
(23)
Since $BU^* = V$, from the hypothesis (15), we have
\[ \left\| p^T \hat{p} \Delta^{(k)} - p^T pX^{(k)} \right\|_F < \tau_k \| p^T \|_F \left\| U^* \right\|_F \leq \tau_k \left\| p^T \right\|_F \left\| \hat{U}^{(k)} - U^* \right\|_F. \]
(24)
Combining (23) with (24), we have
\[ \left\| \hat{U}^{(k+1)} - U^* \right\|_F \leq \left\| \left( I - P^{-1}B \right) \left( \hat{U}^{(k)} - U^* \right) \right\|_F + \left\| p^{-1} \left\| P^{-T} \left\| p^T \| pX^{(k)} - p^T \hat{p} \Delta^{(k)} \right\|_F \right\|_F \leq \left( \left\| I - P^{-1}B \right\|_F + \tau_k \left\| P^{-1} \right\|_F \left\| P^{-T} \right\|_F \left\| U^* \right\|_F \right) \left\| \hat{U}^{(k)} - U^* \right\|_F. \]
(25)
Since $\left\| I - P^{-1}B \right\|_F < 1$, we have from (25) that $\tau_{\max} < 1$ for small enough $\tau_{\max}$. The proof is thus complete. \qed

Theorem 1 tells us that the error of the IPPIA iteration consists of two parts: $\left\| I - P^{-1}B \right\|_F$ and $\tau_k \left\| P^{-1} \right\|_F \left\| P^{-T} \right\|_F \left\| U^* \right\|_F$. The first part is the error upper bound for PPIA iteration. The second part is the error upper bound resulting from the CG method, depending on the choice of the stopping tolerance $\tau_k$. This $\tau_k$ is hard to be optimally determined between the convergence rate and computational costs in practice.

The following theorem presents one possible way of choosing the tolerance $\tau_k$.

**Theorem 2.** Let the assumptions in Theorem 1 be satisfied. Suppose that $\{\zeta_k\}$ is a nondecreasing and nonnegative sequence satisfying $\zeta^{(k)} \geq 1$, and $\limsup_{k \to \infty} \zeta^{(k)} = +\infty$, and that $\chi$ is a real constant in the interval $(0, 1)$ satisfying
\[ \tau_k \leq c\chi^{\zeta^{(k)}}, k = 0, 1, \ldots, \]
(26)
where $c$ is a nonnegative constant. Then we have
\[ \limsup_{k \to \infty} \frac{\left\| \hat{U}^{(k+1)} - U^* \right\|_F}{\left\| \hat{U}^{(k)} - U^* \right\|_F} = \left\| I - P^{-1}B \right\|_F, \]
i.e., the convergence rate of IPPIA is asymptotically the same as that of PPIA.

**Proof.** From (25) and (26), we have
\[ \left\| \hat{U}^{(k+1)} - U^* \right\|_F \leq \left( \left\| I - P^{-1}B \right\|_F + \tau_k \left\| P^{-1} \right\|_F \left\| P^{-T} \right\|_F \left\| U^* \right\|_F \right) \left\| \hat{U}^{(k)} - U^* \right\|_F \leq \left( \left\| I - P^{-1}B \right\|_F + c\chi^{\zeta^{(k)}} \left\| P^{-1} \right\|_F \left\| P^{-T} \right\|_F \left\| U^* \right\|_F \right) \left\| \hat{U}^{(k)} - U^* \right\|_F. \]
Since $\lim_{k \to \infty} c\chi^{\zeta^{(k)}} = 0$, the result follows straightforwardly. \qed

**Remark 3.** Theorem 2 tells us that the convergence rate of IPPIA is asymptotically the same as that of PPIA if the tolerance sequence $\{\tau_k\}$ approaches 0 as $k$ increases. To reduce the computational complexity, it is not necessary for $\tau_k$ to approach zero in actual implementations, which will be shown numerically in the next section.
4. Numerical Experiments

In this section, some numerical examples are given to illustrate the effectiveness of our methods. All the numerical experiments were done on a PC with Intel(R) Core(TM) i5-5200U CPU @2.20 GHz by Matlab R2012b.

In our numerical experiments, we used the Bézier curves and the Said–Ball curves to test the following three examples [12,17,26].

For comparison, we also tested the WPIA and the DWPIA. The parameter values $t_i$ and the optimal weight $\omega$ used in our numerical experiments were the same as those used in [12]; i.e., $t_i = i/n$, for $i = 0, 1, \ldots, n$ and $\omega = 2/(1 + \lambda_n)$ with $\lambda_n$ being the smallest eigenvalue of $B$. Moreover, we set the parameter $a = 1.9$ when we tested the DWPIA.

**Example 1.** Consider the lemniscate of Gerono given by

$$(x(t), y(t)) = (\cos t, \sin t \cos t), \quad t \in \left[-\frac{\pi}{2}, \frac{3\pi}{2}\right].$$

A sequence of $n + 1$ points $\{v_i\}_{i=0}^n$ is taken from the lemniscate of Gerono as follows:

$$v_i = \left( x \left( -\frac{\pi}{2} + i\frac{2\pi}{n} \right), y \left( -\frac{\pi}{2} + i\frac{2\pi}{n} \right) \right), \quad i = 0, 1, \ldots, n.$$  

**Example 2.** Consider the helix of radius 5 given by

$$(x(t), y(t), z(t)) = (5 \cos t, 5 \sin t, t), \quad t \in [0, 6\pi].$$

A sequence of $n + 1$ points $\{v_i\}_{i=0}^n$ is taken from the helix as below

$$v_i = \left( 5 \cos \left( \frac{6\pi}{n} i \right), 5 \sin \left( \frac{6\pi}{n} i \right), \frac{6\pi}{n} i \right), \quad i = 0, 1, \ldots, n.$$  

**Example 3.** Consider the four-leaf clover given by

$$(x(t), y(t)) = (4 \sin (t \sin (4t)), 4 \cos (t \sin (4t))), \quad t \in [0, 2\pi].$$

A sequence of $n + 1$ points $\{v_i\}_{i=0}^n$ is taken from the four-leaf clover in the following way

$$v_i = \left( 4 \sin \left( \frac{2\pi}{n} i \right) \sin \left( \frac{8\pi}{n} i \right), 4 \cos \left( \frac{2\pi}{n} i \right) \sin \left( \frac{8\pi}{n} i \right) \right), \quad i = 0, 1, \ldots, n.$$  

4.1. Tests for PPIA

Numerically, we found that the half-bandwidth of the preconditioner $P$ in (8) $q \approx n/2$ is a good choice. Therefore, when we used Bézier curves and Said–Ball curves to test Example 1 with $n = 10$, we took $q = 5$ and 6, respectively. Similarly, when we used Bézier curves and Said–Ball curves to test Example 2 with $n = 18$, we took $q = 9$ and 12, respectively; when we used Bézier curves and Said–Ball curves to test Example 3 with $n = 21$, we took $q = 11$ and 14, respectively.

The spectral distributions of $I - B$ and $I - P^{-1}B$ resulting from Bézier curves and Said–Ball curves in Examples 1–3 are illustrated in Figures 1–3, respectively. From those figures, we can see that most of the eigenvalues of the preconditioned iteration matrices $I - P^{-1}B$ are far away from 1 and clustered around 0 in each of six cases.
To further illustrate the effectiveness of the PPIA, we list in Table 1 the spectral radii of the iteration matrices of the WPIA, DWPIA and PPIA formats resulting from the considered Bézier and Said–Ball curves. Clearly, the spectral radii of the iteration matrices of the PPIA are much less than those of the corresponding WPIA, DWPIA. Additionally, we see that the spectral radii of PPIA decline sharply when the half-bandwidth of the preconditioner $P$ is near $n/2$. Those results entirely coincide with the aforementioned conclusions in Section 2. Therefore, we can expect that the PPIA method has a good convergence behavior.

We denote the number of iterations by $k$, the $k$-th interpolation error by $\varepsilon^{(k)}$ and the elapsed CPU time after $k$ iterations by $t^{(k)}$ (in seconds), respectively. We emphasize here that the elapsed CPU time for the same routine in Matlab was different every time. Therefore, we took the average of all runtimes of 100 independent runs as the elapsed CPU time. The detailed numerical results of the PPIA, DWPIA and WPIA resulting from Bézier curves and Said–Ball curves are listed in Tables 2–4, respectively. Under the requirement of the same precision, the number of iterations by PPIA is much less than those of WPIA and DWPIA, and the runtime by PPIA is much less than the runtimes of WPIA and DWPIA. This means that our PPIA can accelerate the convergence rate of PIA significantly.

In Table 5, we show the detailed runtimes of WPIA, DWPIA and PPIA for when we tested Example 3 by using Bézier curves and Said–Ball curves. We denote the runtime of computing the optimal weight for WPIA by $T_1$, the runtime of computing different weights for DWPIA by $T_2$ and the runtime of constructing the preconditioner $P$ for PPIA by $T_3$, respectively. We can see that the cost of constructing $P$ is small enough and can be neglected.
Table 1. The spectral radii of iteration matrices of WPIA, DWPIA and PPIA in Examples 1–3.

|          | Bézier Curve | Said–Ball Curve |
|----------|--------------|-----------------|
|          | WPIA         | DWPIA           | PPIA | WPIA         | DWPIA           | PPIA |
| Example 1| 0.99927450   | 0.99924155      | 0.16473699 | 0.99964097   | 0.99947135      | 0.53425173 |
| Example 2| 0.99999967   | 0.99999996      | 0.80078821 | 0.99999992   | 0.99999988      | 0.82319850 |
| Example 3| 1.00000000   | 0.99999998      | 0.72221739 | 1.00000000   | 1.00000000      | 0.98124671 |

Table 2. The approximation errors and runtimes of WPIA, DWPIA and PPIA in Example 1.

| Curve     | k   | $\epsilon^{(k)}$ | $t^{(k)}$ | $\epsilon^{(k)}$ | $t^{(k)}$ | $\epsilon^{(k)}$ | $t^{(k)}$ |
|-----------|-----|-----------------|---------|-----------------|---------|-----------------|---------|
| Bézier    | 0   | $4.620 \times 10^{-1}$ | –       | $4.620 \times 10^{-1}$ | –       | $4.620 \times 10^{-1}$ | –       |
|           | 1   | $2.632 \times 10^{-1}$ | $8.29 \times 10^{-4}$ | $2.601 \times 10^{-1}$ | $4.83 \times 10^{-5}$ | $7.500 \times 10^{-4}$ | $8.67 \times 10^{-5}$ |
|           | 10  | $5.447 \times 10^{-2}$ | $8.96 \times 10^{-4}$ | $5.208 \times 10^{-2}$ | $8.94 \times 10^{-5}$ | $5.017 \times 10^{-3}$ | $1.42 \times 10^{-4}$ |
|           | 411 | $8.611 \times 10^{-4}$ | $2.65 \times 10^{-3}$ | $7.510 \times 10^{-4}$ | $1.83 \times 10^{-3}$ | –               | –       |
|           | 430 | $7.499 \times 10^{-4}$ | $2.80 \times 10^{-3}$ | –               | –       | –               | –       |
| Said–Ball | 0   | $5.470 \times 10^{-1}$ | –       | $5.470 \times 10^{-1}$ | –       | $5.470 \times 10^{-1}$ | –       |
|           | 1   | $4.143 \times 10^{-1}$ | $8.42 \times 10^{-4}$ | $3.414 \times 10^{-1}$ | $4.79 \times 10^{-5}$ | $4.048 \times 10^{-3}$ | $8.62 \times 10^{-5}$ |
|           | 10  | $8.979 \times 10^{-2}$ | $9.13 \times 10^{-4}$ | $5.524 \times 10^{-2}$ | $8.79 \times 10^{-5}$ | $1.733 \times 10^{-3}$ | $1.56 \times 10^{-4}$ |
|           | 20  | $5.252 \times 10^{-2}$ | $9.27 \times 10^{-4}$ | $2.909 \times 10^{-2}$ | $1.32 \times 10^{-4}$ | $2.257 \times 10^{-3}$ | $2.11 \times 10^{-4}$ |
|           | 310 | $6.058 \times 10^{-3}$ | $2.20 \times 10^{-3}$ | $4.045 \times 10^{-3}$ | $1.43 \times 10^{-3}$ | –               | –       |
|           | 478 | $4.051 \times 10^{-3}$ | $2.99 \times 10^{-3}$ | –               | –       | –               | –       |

Table 3. The approximation errors and runtime of WPIA, DWPIA and PPIA in Example 2.

| Curve     | k   | $\epsilon^{(k)}$ | $t^{(k)}$ | $\epsilon^{(k)}$ | $t^{(k)}$ | $\epsilon^{(k)}$ | $t^{(k)}$ |
|-----------|-----|-----------------|---------|-----------------|---------|-----------------|---------|
| Bézier    | 0   | $4.625 \times 10^{0}$ | –       | $4.625 \times 10^{0}$ | –       | $4.625 \times 10^{0}$ | –       |
|           | 1   | $3.868 \times 10^{0}$ | $1.08 \times 10^{-3}$ | $3.866 \times 10^{0}$ | $6.19 \times 10^{-5}$ | $8.396 \times 10^{-5}$ | $1.20 \times 10^{-4}$ |
|           | 10  | $9.821 \times 10^{-1}$ | $1.26 \times 10^{-3}$ | $1.001 \times 10^{0}$ | $1.15 \times 10^{-4}$ | $1.943 \times 10^{-6}$ | $2.12 \times 10^{-4}$ |
|           | 40  | $2.014 \times 10^{-1}$ | $1.38 \times 10^{-3}$ | $2.012 \times 10^{-1}$ | $3.55 \times 10^{-4}$ | $2.858 \times 10^{-9}$ | $5.47 \times 10^{-4}$ |
|           | 163,447 | $8.417 \times 10^{-5}$ | $1.22 \times 10^{0}$ | $8.396 \times 10^{-5}$ | $1.24 \times 10^{0}$ | –               | –       |
|           | 163,902 | $8.396 \times 10^{-5}$ | $1.24 \times 10^{0}$ | –               | –       | –               | –       |
| Said–Ball | 0   | $5.698 \times 10^{0}$ | –       | $5.698 \times 10^{0}$ | –       | $5.698 \times 10^{0}$ | –       |
|           | 1   | $6.614 \times 10^{0}$ | $1.19 \times 10^{-3}$ | $5.585 \times 10^{0}$ | $6.15 \times 10^{-5}$ | $3.281 \times 10^{-4}$ | $1.23 \times 10^{-4}$ |
|           | 10  | $1.812 \times 10^{0}$ | $1.30 \times 10^{-3}$ | $1.822 \times 10^{0}$ | $1.19 \times 10^{-4}$ | $7.809 \times 10^{-6}$ | $2.21 \times 10^{-4}$ |
|           | 40  | $3.319 \times 10^{-1}$ | $1.37 \times 10^{-3}$ | $2.429 \times 10^{-1}$ | $3.24 \times 10^{-4}$ | $2.078 \times 10^{-8}$ | $5.60 \times 10^{-4}$ |
|           | 72,224 | $3.281 \times 10^{-4}$ | $5.36 \times 10^{-1}$ | $3.358 \times 10^{-4}$ | $5.31 \times 10^{-1}$ | –               | –       |
|           | 74,606 | –               | –       | $3.281 \times 10^{-4}$ | $5.97 \times 10^{-1}$ | –               | –       |
Table 4. The approximation errors and runtime of WPIA, DWPIA and PPIA in Example 3.

| Curve      | $k$ | $\epsilon^{(k)}$ | $t^{(k)}$ | $\epsilon^{(k)}$ | $t^{(k)}$ | $\epsilon^{(k)}$ | $t^{(k)}$ |
|------------|-----|------------------|-----------|------------------|-----------|------------------|-----------|
| Bézier     | 0   | $4.625 \times 10^0$ | –         | $4.625 \times 10^0$ | –         | $4.625 \times 10^0$ | –         |
|            | 1   | $9.493 \times 10^{-1}$ | $9.54 \times 10^{-4}$ | $9.487 \times 10^{-1}$ | $5.39 \times 10^{-5}$ | $6.363 \times 10^{-7}$ | $1.20 \times 10^{-4}$ |
|            | 10  | $6.657 \times 10^{-2}$ | $1.14 \times 10^{-3}$ | $7.524 \times 10^{-2}$ | $9.74 \times 10^{-5}$ | $2.104 \times 10^{-10}$ | $1.95 \times 10^{-4}$ |
|            | 40  | $6.615 \times 10^{-3}$ | $1.30 \times 10^{-3}$ | $6.903 \times 10^{-3}$ | $2.67 \times 10^{-4}$ | $1.245 \times 10^{-14}$ | $4.21 \times 10^{-4}$ |
|            | 49,109 | $6.363 \times 10^{-7}$ | $2.34 \times 10^{-1}$ | $6.405 \times 10^{-7}$ | $2.39 \times 10^{-1}$ | – | – |
|            | 49,369 | – | – | $6.363 \times 10^{-7}$ | $2.44 \times 10^{-1}$ | – | – |
| Said–Ball  | 0   | $4.049 \times 10^0$ | –         | $4.049 \times 10^0$ | –         | $4.049 \times 10^0$ | –         |
|            | 1   | $2.375 \times 10^0$ | $9.30 \times 10^{-4}$ | $3.176 \times 10^0$ | $7.08 \times 10^{-5}$ | $2.127 \times 10^{-5}$ | $1.25 \times 10^{-4}$ |
|            | 10  | $6.678 \times 10^{-2}$ | $1.11 \times 10^{-3}$ | $4.639 \times 10^{-1}$ | $1.06 \times 10^{-4}$ | $1.638 \times 10^{-7}$ | $1.88 \times 10^{-4}$ |
|            | 40  | $2.519 \times 10^{-2}$ | $1.25 \times 10^{-3}$ | $3.202 \times 10^{-2}$ | $2.57 \times 10^{-4}$ | $7.217 \times 10^{-8}$ | $4.24 \times 10^{-4}$ |
|            | 2776 | $4.285 \times 10^{-5}$ | $1.44 \times 10^{-2}$ | $2.128 \times 10^{-5}$ | $1.39 \times 10^{-2}$ | – | – |
|            | 5354 | $2.127 \times 10^{-5}$ | $2.60 \times 10^{-2}$ | – | – | – | – |

Table 5. The detailed runtime of WPIA, DWPIA and PPIA in Example 3.

| Curve      | $T_1$ | $T_2$ | $T_3$ |
|------------|-------|-------|-------|
| Bézier     | $1.12 \times 10^{-3}$ | $1.83 \times 10^{-5}$ | $2.75 \times 10^{-5}$ |
| Said–Ball  | $1.06 \times 10^{-3}$ | $1.70 \times 10^{-5}$ | $2.48 \times 10^{-5}$ |

We remark here that all curves in Figures 4–6 are those obtained at 10th iteration step by employing different iterations with different bases, respectively. In Figures 4–6, the curves generated by PPIA and WPIA formats are denoted by the solid and dashed lines respectively. We can see from Figures 4–6 that the approximations obtained by PPIA are better than those obtained by WPIA, no matter whether Bézier curves or Said–Ball curves are used. Clearly, the PPIA converges faster than the WPIA.

![Figure 4. Approximations of the lemniscate of Gerono in Example 1 at the tenth iteration.](image-url)
4.2. Tests for IPPIA

From the above numerical experiments, we know that the PPIA works much better than the WPIA and DWPIA if the number of the interpolating points is moderate. However, when we fit 53 points sampled from the lemniscate of Gerono, the helix and the four-leaf clover in Examples 1–3 by using Bézier curves and Said–Ball curves, we found that the WPIA and DWPIA converge very slowly. At the same time, we found that the PPIA does not work, because the condition number of $P$ is very large and the solution to $P\Delta^{(k)} = \hat{\Delta}^{(k)}$ cannot be directly obtained. In this case, we can employ the IPPIA to get the corresponding fitting curves. The numerical results (including the number of iterations, the approximation errors and the runtime) obtained by the WPIA, DWPIA and IPPIA are listed in Tables 6–8 respectively, where the parameters in IPPIA are taken as $q = 26$, $\theta = 0.98$ and $\tau_k = 10^{-2}$.

From Tables 6–8, we can see that the IPPIA converges much faster than the WPIA and DWPIA. For example, when we fit the points sampled from the lemniscate of Gerono by using Bézier curves, we observed that the errors caused by performing 5 IPPIA iterations are about the same as those caused by performing 137 WPIA iterations or 381 DWPIA iterations. Again, when we fit 53 points sampled from the lemniscate of Gerono by using Said–Ball curves, we found that the errors caused by performing 4 IPPIA iterations are about the same as those caused by performing 291 WPIA iterations or 358 DWPIA iterations. Furthermore, under the requirement of the same accuracy, the runtime of IPPIA iterations was less than that of WPIA iterations and it was comparable to that of DWPIA iterations.

When fitting 53 points sampled from the lemniscates of Gerono, helix and four-leaf clover, we sketched the Bézier curves and the Said–Ball curves generated by performing five IPPIA and WPIA iterations in Figures 7–9, respectively. The solid and dashed lines in Figures 7–9 represent the curves generated by IPPIA and WPIA respectively. Clearly, the IPPIA outperforms the WPIA in the sense of the convergence rate.

Figure 5. Approximations of the helix in Example 2 at the tenth iteration.

Figure 6. Approximations of the four-leaf clover in Example 3 at the tenth iteration.
Table 6. Numerical results when fitting 53 points sampled from the lemniscate of Gerono.

| Curve     | k  | WPIA $t(x)$ | DWPIA $t(x)$ | IPPIA $t(x)$ |
|-----------|----|-------------|--------------|--------------|
| Bézier    | 0  | 1.606 × 10^{-1} | – | 1.606 × 10^{-1} | – | 1.606 × 10^{-1} | – |
|           | 1  | 7.987 × 10^{-2} | – | 7.289 × 10^{-2} | – | 1.507 × 10^{-3} | – |
|           | 3  | 4.125 × 10^{-2} | – | 3.313 × 10^{-2} | – | 4.537 × 10^{-7} | – |
|           | 5  | 3.018 × 10^{-2} | – | 2.126 × 10^{-2} | – | 3.78 × 10^{-9}  | 2.39 × 10^{-3}|
|          | 137 | 3.451 × 10^{-9} | 2.66 × 10^{-3} | 1.192 × 10^{-8} | – | – | – |
|          | 233 | 1.336 × 10^{-1} | – | 3.456 × 10^{-9} | 2.94 × 10^{-3} | – | – |
| Said–Ball | 0  | 4.308 × 10^{-1} | – | 4.308 × 10^{-1} | – | 4.308 × 10^{-1} | – |
|           | 1  | 3.789 × 10^{-1} | – | 4.225 × 10^{-1} | – | 1.265 × 10^{-2} | – |
|           | 2  | 3.602 × 10^{-1} | – | 3.139 × 10^{-1} | – | 2.369 × 10^{-4} | – |
|           | 3  | 9.127 × 10^{-2} | – | 1.305 × 10^{-1} | – | 9.892 × 10^{-6} | – |
|           | 4  | 8.820 × 10^{-2} | – | 1.360 × 10^{-1} | – | 3.396 × 10^{-6} | 2.42 × 10^{-3}|
|          | 291 | 3.382 × 10^{-6} | 3.43 × 10^{-3} | 6.918 × 10^{-6} | – | – | – |
|          | 358 | –            | – | 3.382 × 10^{-6} | 2.82 × 10^{-3} | – | – |

Figure 7. Approximations to 53 points sampled from the lemniscate of Gerono at the fifth iteration.

Table 7. Numerical results when fitting 53 points sampled from the helix.

| Curve     | k  | WPIA $t(x)$ | DWPIA $t(x)$ | IPPIA $t(x)$ |
|-----------|----|-------------|--------------|--------------|
| Bézier    | 0  | 2.882 × 10^{0} | – | 2.882 × 10^{0} | – | 2.882 × 10^{0} | – |
|           | 1  | 7.021 × 10^{-1} | – | 6.469 × 10^{-1} | – | 5.007 × 10^{-2} | – |
|           | 2  | 3.359 × 10^{-1} | – | 2.827 × 10^{-1} | – | 5.533 × 10^{-4} | – |
|           | 4  | 1.997 × 10^{-1} | – | 1.506 × 10^{-1} | – | 6.341 × 10^{-7} | 3.03 × 10^{-3}|
|          | 233 | 1.536 × 10^{-5} | – | 6.349 × 10^{-7} | 2.56 × 10^{-3} | – | – |
|          | 316 | 6.416 × 10^{-7} | 4.23 × 10^{-3} | – | – | – | – |
| Said–Ball | 0  | 5.773 × 10^{0} | – | 5.773 × 10^{0} | – | 5.773 × 10^{0} | – |
|           | 1  | 6.681 × 10^{0} | – | 5.763 × 10^{0} | – | 2.508 × 10^{-1} | – |
|           | 3  | 4.316 × 10^{0} | – | 5.213 × 10^{0} | – | 2.913 × 10^{-4} | – |
|           | 5  | 2.086 × 10^{0} | – | 2.062 × 10^{0} | – | 9.780 × 10^{-5} | 3.26 × 10^{-3}|
|          | 359 | 2.110 × 10^{-4} | – | 9.750 × 10^{-5} | 3.84 × 10^{-3} | – | – |
|          | 438 | 9.765 × 10^{-5} | 5.40 × 10^{-3} | – | – | – | – |
Figure 8. Approximations of 53 points sampled from the helix at the fifth iteration.

Table 8. Numerical results when fitting 53 points sampled from the four-leaf clover.

| Curve     | $k$ | WPIA $\epsilon(k)$ | WPIA $t(k)$ | DWPIA $\epsilon(k)$ | DWPIA $t(k)$ | IPPIA $\epsilon(k)$ | IPPIA $t(k)$ |
|-----------|-----|----------------------|-------------|----------------------|-------------|----------------------|-------------|
| Bézier    | 0   | $1.219 \times 10^0$  | –           | $1.219 \times 10^0$  | –           | $1.219 \times 10^0$  | –           |
|           | 1   | $3.815 \times 10^{-1}$ | –           | $3.460 \times 10^{-1}$ | –           | $1.967 \times 10^{-2}$ | –           |
|           | 2   | $2.270 \times 10^{-1}$ | –           | $1.915 \times 10^{-1}$ | –           | $1.526 \times 10^{-4}$ | –           |
|           | 4   | $1.226 \times 10^{-1}$ | –           | $9.344 \times 10^{-2}$ | –           | $2.802 \times 10^{-7}$ | –           |
|           | 125 | $1.378 \times 10^{-4}$ | –           | $2.791 \times 10^{-7}$ | 1.18 $\times 10^{-3}$ |                | –           |
|           | 316 | $2.799 \times 10^{-7}$ | 3.60 $\times 10^{-3}$ | –                     | –           |                       | –           |
| Said–Ball | 0   | $3.338 \times 10^0$  | –           | $3.338 \times 10^0$  | –           | $3.338 \times 10^0$  | –           |
|           | 1   | $2.227 \times 10^0$  | –           | $3.025 \times 10^0$  | –           | $9.302 \times 10^{-2}$ | –           |
|           | 2   | $2.271 \times 10^0$  | –           | $3.543 \times 10^0$  | –           | $4.527 \times 10^{-3}$ | –           |
|           | 4   | $4.344 \times 10^{-1}$ | –           | $1.495 \times 10^0$  | –           | $3.743 \times 10^{-5}$ | 2.58 $\times 10^{-3}$ |
|           | 218 | $3.741 \times 10^{-5}$ | 2.99 $\times 10^{-3}$ | $1.505 \times 10^{-4}$ | –           | –                     | –           |
|           | 379 | –                     | –           | $3.712 \times 10^{-5}$ | 3.02 $\times 10^{-3}$ | –                     | –           |

Figure 9. Approximations of 53 points sampled from the four-leaf clover at the fifth iteration.

5. Conclusions

In this paper, we have exploited the properties of collocation matrix $B$, resulting from an NTP basis, to design an efficient preconditioner $P$ for PIA. The main tool to derive our preconditioner is the diagonally compensated reduction. By applying that $P$ to PIA, we have developed a novel version of PIA, called the PPIA. By both convergence theory and numerical experiments, we have shown that the PPIA outperforms the WPIA and DWPIA, in the sense of convergence rate and elapsed CPU...
time. In particular, to reduce the computational complexity of PPIA, we have also proposed an inexact variant of PPIA (i.e., IPPIA) and shown its effectiveness, especially when the number of interpolating points is large.

As mentioned in Section 1, the PIA has extensive successful applications. Therefore, we can expect that our methods have a wide range of potential applications.

**Author Contributions:** Methodology, Z.L.; Resources, C.L.; Writing—original draft, C.L.; Writing—review and editing, Z.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Natural Science Foundation of China (grant numbers 11371075 and 11771453), the Hunan Key Laboratory of mathematical modeling and analysis in engineering, Natural Science Foundation of Hunan Province (grant number 2020JJ5267) and the Scientific Research Funds of Hunan Provincial Education Department (grant number 18C877).

**Conflicts of Interest:** The authors declare no conflicts of interest.

**References**

1. Hu, Q. An iterative algorithm for polynomial of rational triangular Bézier surfaces. *Appl. Math. Comput.* 2013, 219, 9308–9316. [CrossRef]
2. Gofuku, S.; Tamura, S.; Maekawa, T. Point-tangent/point-normal B-spline curve interpolation by geometric algorithms. *Comput. Aided Des.* 2009, 41, 412–422. [CrossRef]
3. Lin, H. Adaptive data fitting by the progressive-iterative approximation. *Comput. Aided Geom. Des.* 2012, 29, 463–473. [CrossRef]
4. Lin, H.; Zhang, Z. An efficient method for fitting large data sets using T-splines. *SIAM J. Sci. Comput.* 2013, 35, 3052–3068. [CrossRef]
5. Kineri, Y.; Wang, M.; Lin, H.; Maekawa, T. B-spline surface fitting by iterative geometric interpolation/approximation algorithms. *Comput. Aided Des.* 2012, 44, 697–708. [CrossRef]
6. Lin, H.; Jin, S.; Liao, H.; Jian, Q. Quality guaranteed all-hex mesh generation by a constrained volume iterative fitting algorithm. *Comput. Aided Des.* 2015, 67, 107–117. [CrossRef]
7. Martin, T.; Cohen, E.; Kirby, R. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. *Comput. Aided Geom. Des.* 2009, 26, 648–664. [CrossRef]
8. Lin, H.; Jin, S.; Hu, Q.; Liu, Z. Constructing B-spline solids from tetrahedral meshes for isogeometric analysis. *Comput. Aided Geom. Des.* 2015, 3, 109–120. [CrossRef]
9. Lin, H.; Maekawa, T.; Deng, C. Survey on geometric iterative methods and their applications. *Comput. Aided Des.* 2017, 95, 40–51. [CrossRef]
10. Marco, A.; Martínez, J. A fast and accurate algorithm for solving Bernstein-Vandermonde linear systems. *Linear Algebra Appl.* 2006, 2, 616–628. [CrossRef]
11. Deng, S.; Wang, G. Numerical analysis of the progressive iterative approximation method. *Comput. Aided Des. Graph.* 2012, 7, 879–884.
12. Lu, L. Weighted progressive iteration approximation and convergence analysis. *Comput. Aided Geom. Des.* 2010, 2, 129–137. [CrossRef]
13. Carnicer, J.M.; Delgado, J.; Pena, J. Richardson method and totally nonnegative linear systems. *Linear Algebra Appl.* 2010, 11, 2010–2017. [CrossRef]
14. Liu, X.; Deng, C. Jacobi-PIA algorithm for non-uniform cubic B-Spline curve interpolation. *J. Comput. Aided Des. Comput. Graph.* 2015, 27, 485–491.
15. Liu, C.; Han, X.; Li, J. The Chebyshev accelerating method for progressive iterative approximation. *Commun. Inf. Syst.* 2017, 17, 25–43. [CrossRef]
16. Zhang, L.; Zhao, L.; Tan, J. Progressive iterative approximation with different weights and its application. *J. Zhejiang Univ. Sci. Ed.* 2017, 44, 22–27.
17. Wang, Z.; Li, J.; Deng, C. Convergence proof of GS-PIA algorithm. *J. Comput. Aided Des. Comput. Graph.* 2018, 30, 60–66. [CrossRef]
18. Zhang, L.; Tan, J.; Ge, X.; Guo, Z. Generalized B-splines’ geometric iterative fitting method with mutually different weights. *J. Comput. Appl. Math.* 2018, 329, 331–343. [CrossRef]
19. Ebrahimi, A.; Loghmani, G. A composite iterative procedure with fast convergence rate for the progressive-iteration approximation of curves. *J. Comput. Appl. Math.* 2019, 359, 1–15. [CrossRef]
20. Lin, H. Local progressive-iterative approximation format for blending curves and patches. *Comput. Aided Geom. Des.* 2010, 27, 322–339. [CrossRef]
21. Lin, H.; Zhang, Z. An extended iterative format for the progressive-iteration approximation. *Comput. Graph.* 2011, 35, 967–975. [CrossRef]
22. Deng, C.; Lin, H. Progressive and iterative approximation for least squares B-spline curve and surface fitting. *Comput. Aided Des.* 2014, 47, 32–44. [CrossRef]
23. Farouki, R. The Bernstein polynomial basis: A centennial retrospective. *Comput. Aided Geom. Des.* 2012, 29, 379–419. [CrossRef]
24. Delgado, J.; Pena, J. On the generalized Ball bases. *Adv. Comput. Math.* 2006, 24, 263–280. [CrossRef]
25. Lin, H.; Bao, H.; Wang, G. Totally positive bases and progressive iteration approximation. *Comput. Math. Appl.* 2005, 50, 575–586. [CrossRef]
26. Carnicer, J.M.; Delgado, J. On the progressive iteration approximation property and alternative iterations. *Comput. Aided Geom. Des.* 2011, 28, 523–526. [CrossRef]
27. Liu, Z.; Wu, N.; Qin, X.; Zhang, Y. Trigonometric transform splitting methods for real symmetric Toeplitz systems. *Comput. Math. Appl.* 2018, 75, 2782–2794. [CrossRef]
28. Liu, Z.; Qin, X.; Wu, N.; Zhang, Y. The shifted classical circulant and skew circulant splitting iterative methods for Toeplitz matrices. *Canad. Math. Bull.* 2017, 60, 807–815. [CrossRef]
29. Liu, Z.; Ralha, R.; Zhang, Y.; Ferreira, C. Minimization problems for certain structured matrices. *Electron. J. Linear Algebra* 2015, 30, 613–631. [CrossRef]
30. Axelsson, O. *Iterative Solution Methods*; Cambridge University Press: Cambridge, UK, 1994.
31. Cao, Z.; Liu, Z. Symmetric multisplitting of a symmetric positive definite matrix. *Linear Algebra Appl.* 1998, 285, 309–319. [CrossRef]