Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 9940610, 12 pages
https://doi.org/10.1155/2021/9940610

Research Article

PIA and PPIA for Interpolating Points and Derivatives at Endpoints by Bézier Curves

Yeqing Yi,[1] **Lijuan Hu**,[2] **and Chengzhi Liu**[2]

[1]*School of Information, Hunan University of Humanities, Science and Technology, Loudi 417000, China*
[2]*School of Mathematics and Finance, Hunan University of Humanities, Science and Technology, Loudi 417000, China*

Correspondence should be addressed to Chengzhi Liu; 162101002@csu.edu.cn

Received 9 March 2021; Accepted 14 June 2021; Published 1 July 2021

Academic Editor: Gaetano Giunta

In this study, we are concerned with the interpolation problem that interpolates not only a given set of points but also derivatives at endpoints by using Bézier curves. The progressive iterative approximation (PIA) is proposed to interpolate the given points and derivatives. To speed up the convergence rate of PIA, we exploit the preconditioned PIA (PPIA), in which the diagonally compensated reduction is used to construct the preconditioner. The convergence of PIA and PPIA is also analyzed in this study. The proposed PPIA is applied to approximate higher order or/and rational Bézier curves. Numerical examples are given to illustrate the effectiveness of the proposed methods.

1. Introduction

We begin with the description of the data interpolation problem with constrains at endpoints. Given a set of organized points $\{\mathbf{p}_i\}_{i=0}^{n} \in \mathbb{R}^2$ or $\mathbb{R}^3$, r^{th} derivatives $\mathbf{d}_0^r (r = 1, 2, \ldots, u) \in \mathbb{R}^2$ or $\mathbb{R}^3$ at the endpoint $\mathbf{p}_0$ and s^{th} derivatives $\mathbf{d}_n^s (s = 1, 2, \ldots, v) \in \mathbb{R}^2$ or $\mathbb{R}^3$ at the endpoint $\mathbf{p}_n$. For $i = 0, 1, \ldots, n$, the i^{th} point $\mathbf{p}_i$ assigned an ordered parameter t_i, i.e., $t_0 < t_1 < \cdots < t_n$. We want to find a m degree Bézier curve,

$$\mathbf{C}(t) = \sum_{j=0}^{m} \mathbf{q}_j B_j^m(t), \tag{1}$$

that interpolates these points as well as the derivatives at the endpoints, i.e.,

$$\begin{cases} \mathbf{C}(t_i) = \mathbf{p}_i, & i = 0, 1, \ldots, n; \\ \dfrac{\mathrm{d}^r \mathbf{C}(t)}{\mathrm{d}t^r}\Big|_{t=t_0} = \mathbf{d}_0^r, & r = 1, 2, \ldots, u; \\ \dfrac{\mathrm{d}^s \mathbf{C}(t)}{\mathrm{d}t^s}\Big|_{t=t_n} = \mathbf{d}_n^s, & s = 1, 2, \ldots, v. \end{cases} \tag{2}$$

The terms $B_j^m(t)$ in (1) are the Bernstein polynomials of degree m, i.e., $B_j^m(t) = t^j (1-t)^{m-j}$. According to the existence and uniqueness of polynomial interpolant, since there are $n + u + v + 1$ different interpolation conditions in (2), we have $m = n + u + v$.

Data interpolation and interpolation with tangents, curvatures, and other derivatives appear frequently in the fields of science and engineering [1, 2]. It is difficult to represent curves with complex shapes by a single curve, and a composite curve composed of several pieces of curves is necessary [3]. Hence, we need to control the smoothness of the composite curve when piecing curves together. It would be great if the derivatives at the endpoints are known. Very often, this knowledge is not given in practice, and we have to determine the derivatives at the endpoints. A careful choice of end conditions is important because it determines the shape of the interpolating curve near the endpoints. There are several strategies, e.g., natural end condition, Bessel end condition, quadratic end condition [4, 5]. In this study, we are especially interested in exploiting iterative methods for solving the interpolation curve (1) and do not consider the determination of constraints at the endpoints.

It is known to all that the interpolation Bézier curve (1) can be obtained by solving a system of linear equations directly when the coefficient matrix is well-conditioned. However, the condition number of the linear system will be very large for large m; therefore, it is required to introduce an efficient algorithm to solve the linear system. In recent years, a geometric iterative method, named PIA, plays an important role in data interpolation. Due to its clear geometric meaning, stable convergence, and simple iterative scheme, PIA has intrigued the researchers for decades. For more details about PIA, readers can refer to a recent survey study [6], in which the authors summarized PIAs and their successful applications.

In this study, we exploit the PIA format that interpolates not only a given set of points but also derivatives at endpoints by using Bézier curves. The study is organized as follows. In Section 2, we propose the PIA format and then analyze its convergence. In order to accelerate the convergence rate of PIA, we exploit the preconditioning technique for PIA in Section 3. In Section 4, the proposed method is applied to approximate higher order or rational Bézier curves with constraints. Numerical examples are given to illustrate the effectiveness of our proposed methods in Section 5, and some conclusion remarks are given in the last section.

2. Progressive Iterative Approximation

2.1. Interpolating Derivatives Conditions at Endpoints. We note that the k^{th} derivative of (1) is given by

$$\frac{\mathrm{d}^k \mathbf{C}(t)}{\mathrm{d}t^k} = \frac{m!}{(m-k)!} \sum_{j=0}^{m-k} \Delta^k \mathbf{q}_j B_j^{m-k}(t), \tag{3}$$

where $\Delta^k \mathbf{q}_j$ is defined according to the recurrence relative formula:

$$\Delta^k \mathbf{q}_j = \Delta^{k-1} \mathbf{q}_{j+1} - \Delta^{k-1} \mathbf{q}_j, \text{ with } \Delta^0 \mathbf{q}_j = \mathbf{q}_j. \tag{4}$$

By direct calculation, from (2) and (3), we have

$$\begin{cases} \mathbf{C}(0) = \mathbf{q}_0 = \mathbf{p}_0; \\ \mathbf{C}(1) = \mathbf{q}_m = \mathbf{p}_n; \\ \Delta^r \mathbf{q}_0 = \dfrac{(m-r)!}{m!} \mathbf{d}_0^r, \qquad r = 1, 2, \ldots, u; \\ \Delta^s \mathbf{q}_{m-s} = \dfrac{(m-s)!}{m!} \mathbf{d}_n^s, \qquad s = 1, 2, \ldots, v. \end{cases} \tag{5}$$

Hence, the control points $\mathbf{q}_j\,(j = 0, 1, \ldots, u, m-v, \ldots, m)$ in (1) can be determined according to (5). As an example, we outline the control points with respect to the case of $u = v = 3$.

$$\begin{cases} \mathbf{q}_0 = \mathbf{p}_0, \\ \mathbf{q}_m = \mathbf{p}_n, \\ \mathbf{q}_1 = \mathbf{q}_0 + \dfrac{1}{m} \mathbf{d}_0^1, \\ \mathbf{q}_{m-1} = \mathbf{q}_m - \dfrac{1}{m} \mathbf{d}_n^1, \\ \mathbf{q}_2 = \dfrac{(m-2)!}{m!} \mathbf{d}_0^2 + 2\mathbf{q}_1 - \mathbf{q}_0, \\ \mathbf{q}_{m-2} = \dfrac{(m-2)!}{m!} \mathbf{d}_n^2 + 2\mathbf{q}_{m-1} - \mathbf{q}_m, \\ \mathbf{q}_3 = \dfrac{(m-3)!}{m!} \mathbf{d}_0^3 + 3\mathbf{q}_2 - 3\mathbf{q}_1 + \mathbf{q}_0, \\ \mathbf{q}_{m-3} = -\dfrac{(m-3)!}{m!} \mathbf{d}_n^3 + 3\mathbf{q}_{m-2} - 3\mathbf{q}_{m-1} + \mathbf{q}_m. \end{cases} \tag{6}$$

Therefore, the Bézier curve (1) can be expressed by

$$\mathbf{C}(t) = \sum_{j=0}^{u} \mathbf{q}_j B_j^m(t) + \sum_{j=u+1}^{m-v-1} \mathbf{q}_j B_j^m(t) + \sum_{j=m-v}^{m} \mathbf{q}_j B_j^m(t), \tag{7}$$

where the control points $\mathbf{q}_j\,(j = 0, 1, \ldots, u, m-v, \ldots, m)$ can be obtained according to (5), and the control points $\mathbf{q}_j\,(j = u+1, u+2, \ldots, m-v-1)$ need to be determined. We will give a computing method for solving these points in the following subsection.

2.2. Approximate Interpolation Algorithm. First, the given points $\mathbf{p}_i\,(i = 1, 2, \ldots, n-1)$ and $\mathbf{q}_j\,(j = 0, 1, \ldots, u, m-v, \ldots, m)$ are interpreted as the control points of a Bézier curve, and therefore, we can generate an initial approximate interpolation curve:

$$\mathbf{C}^0(t) = \sum_{j=0}^{u} \mathbf{q}_j B_j^m(t) + \sum_{j=u+1}^{m-v-1} \mathbf{q}_j^0 B_j^m(t) + \sum_{j=m-v}^{m} \mathbf{q}_j B_j^m(t), \tag{8}$$

where $\mathbf{q}_j^0 = \mathbf{p}_{j-u}\,(j = u+1, \ldots, m-v-1)$.

Second, for $j = u+1, \ldots, m-v-1$, let $\delta_j^0 = \mathbf{p}_{j-u} - \mathbf{C}^0(t_{j-u})$ be the adjusting vector of the j^{th} control point. Then, we can update the j^{th} point $\mathbf{q}_j^1 = \mathbf{q}_j^0 + \delta_j^0$ and generate the first approximate interpolation curve:

$$\mathbf{C}^1(t) = \sum_{j=0}^{u} \mathbf{q}_j B_j^m(t) + \sum_{j=u+1}^{m-v-1} \mathbf{q}_j^1 B_j^m(t) + \sum_{j=m-v}^{m} \mathbf{q}_j B_j^m(t). \tag{9}$$

We repeat these procedures; thus, we obtain a sequence of approximate interpolation curves:

$$\mathbf{C}^k(t) = \sum_{j=0}^{u} \mathbf{q}_j B_j^m(t) + \sum_{j=u+1}^{m-v-1} \mathbf{q}_j^k B_j^m(t) + \sum_{j=m-v}^{m} \mathbf{q}_j B_j^m(t), \quad k = 1, 2, \ldots, \tag{10}$$

where

$$\mathbf{q}_j^k = \mathbf{q}_j^{k-1} + \boldsymbol{\delta}_j^{k-1}, \boldsymbol{\delta}_j^{k-1} = \mathbf{p}_{j-u} - \mathbf{C}^{k-1}(t_{j-u}), \quad j = u+1, \ldots, m-v-1. \tag{11}$$

Thus, we obtain a sequence of approximate interpolation curves $\mathbf{C}^k(t), k = 0, 1, \ldots,$. The initial Bézier curve is said to have the property of PIA with constraints at endpoints if

$$\begin{cases} \lim_{k\longrightarrow\infty} \mathbf{C}^k(t_i) = \mathbf{p}_i, & i = 0, 1, \ldots, n; \\ \lim_{k\longrightarrow\infty} \dfrac{\mathrm{d}^r \mathbf{C}^k(t)}{\mathrm{d}t^r}\Big|_{t=t_0} = \mathbf{d}_0^r, & r = 1, 2, \ldots, u; \\ \lim_{k\longrightarrow\infty} \dfrac{\mathrm{d}^s \mathbf{C}^k(t)}{\mathrm{d}t^s}\Big|_{t=t_n} = \mathbf{d}_n^s, & s = 1, 2, \ldots, v. \end{cases} \tag{12}$$

Remark 1. By simple calculation, it is easy to verify that these approximate interpolation curves $\mathbf{C}^k(t)$ interpolate the endpoints $\mathbf{p}_0$ and $\mathbf{p}_n$ as well as the derivatives $\mathbf{d}_0^r (r = 1, 2, \ldots, u)$ and $\mathbf{d}_n^s (s = 1, 2, \ldots, v)$.

Let

$$\begin{aligned} \mathbf{Q}^k &= \left[\mathbf{q}_{u+1}^k, \mathbf{q}_{u+2}^k, \ldots, \mathbf{q}_{m-v-1}^k\right]^T, \\ \overline{P} &= \left[\mathbf{p}_1 - \sum_{j=0}^{u} \mathbf{q}_j B_j^m(t_1) - \sum_{j=m-v}^{m} \mathbf{q}_j B_j^m(t_1), \ldots, \mathbf{p}_{m-u-v-1} - \sum_{j=0}^{u} \mathbf{q}_j B_j^m(t_{m-u-v-1}) - \sum_{j=m-v}^{m} \mathbf{q}_j B_j^m(t_{m-u-v-1})\right]^T. \end{aligned} \tag{13}$$

Then, the iterative process (11) can be written in the matrix form:

$$\mathbf{Q}^k = (I - \overline{B})\mathbf{Q}^{k-1} + \overline{P}, \quad k = 1, 2, \ldots, \tag{14}$$

where I is the $(n-1) \times (n-1)$ identity matrix and $\overline{B}$ is said to be the collocation matrix resulting from the Bernstein basis at parameters $\{t_i\}_{i=1}^{n-1}$, in detail

$$\overline{B} = \begin{pmatrix} B_{u+1}^m(t_1) & B_{u+2}^m(t_1) & B_{u+3}^m(t_1) & \cdots & B_{m-v-1}^m(t_1) \\ B_{u+1}^m(t_2) & B_{u+2}^m(t_2) & B_{u+3}^m(t_2) & \cdots & B_{m-v-1}^m(t_2) \\ B_{u+1}^m(t_3) & B_{u+2}^m(t_3) & B_{u+3}^m(t_3) & \cdots & B_{m-v-1}^m(t_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{u+1}^m(t_{n-1}) & B_{r+2}^m(t_{n-1}) & B_{r+3}^m(t_{n-1}) & \cdots & B_{m-v-1}^m(t_{n-1}) \end{pmatrix}. \tag{15}$$

The iterative process (14) is equivalent to the Richardson iteration for solving the system

$$\overline{B}\mathbf{Q} = \overline{P}. \tag{16}$$

2.3. Convergence Analysis of PIA. Before analyzing the convergence, we review some definitions and conclusions.

Definition 1 (see [7, 8]). A matrix is called totally positive if all its minors are positive and positive stable (semistable) if all its eigenvalues have positive (nonnegative) real parts.

It is easy to verify that a positive stable matrix is nonsingular. We note that all the eigenvalues of a totally positive matrix are positive ([9]); thus, a totally positive matrix is also nonsingular.

Definition 2 (see [7]). A basis $\{b_i(t)\}_{i=0}^n$ is called normalized if $b_i(t) \geq 0 (i = 0, \ldots, n)$ and $\sum_{i=0}^n b_i(t) = 1$ and totally positive if its collocation matrix at any increasing sequence is a totally positive matrix.

Lemma 1 (see [8]). *Let A be a given matrix. Then, A is positive stable if and only if for each nonzero vector $\mathbf{x}$, there exists a positive definite matrix K, such that* $\mathrm{Re}(\mathbf{x}^* KA\mathbf{x}) > 0$.

Lemma 2 (see [10]). *Let A be an $n \times n$ matrix with entries a_{ij}. For $i \in \{1, \ldots, n\}$, let $\Lambda_i = \sum_{j\neq i} |a_{ij}|$ be the sum of the absolute values of the nondiagonal entries in the i^{th} row. Let $D(a_{ii}, \Lambda_i) \subseteq \mathbb{C}$ be a closed disc centered at a_{ii} with radius Λ_i. Then, every eigenvalue of A lies within at least one of the discs $D(a_{ii}, \Lambda_i)$.*

Theorem 1. *The Bézier curve has the property of PIA with constraints at endpoints.*

Proof. As stated in Remark 1, the approximate interpolation curves $\mathbf{C}^k(t)$ interpolate endpoints $\mathbf{p}_0$ and $\mathbf{p}_1$ and derivatives $\mathbf{d}_0^r (r = 1, 2, \ldots, u)$ and $\mathbf{d}_n^s (s = 1, 2, \ldots, v)$.

On the other hand, we insert $u + v$ different parameters into the ordered parameters set $\{t_j\}_{j=0}^n$; hence, we obtain a sequence of increasing parameters:

$$t_0 < \tilde{t}_1 < \tilde{t}_2 < \cdots < \tilde{t}_u < t_1 < t_2 < \cdots < t_{n-1} < \bar{t}_1 < \bar{t}_2 < \cdots < \bar{t}_v < t_n. \tag{17}$$

Note that the Bernstein basis is normalized and totally positive [7]. Let B be the collocation matrix resulting from the Bernstein basis at the parameter sequence (17). Then, it follows from Definition 2 that B is totally positive. Since the matrix $\overline{B}$ defined in (15) is a submatrix of B, then $\overline{B}$ is also totally positive according to Definition 1. Hence, all the eigenvalues of $\overline{B}$ are positive.

Again, since the Bernstein basis is normalized, we have $\|B\|_\infty = 1$; thus, the infinite norm of the submatrix $\overline{B}$ is less than 1, i.e., $\|\overline{B}\|_\infty < \|B\|_\infty = 1a$. Therefore, $0 < \lambda_j(\overline{B}) \le \max\{\lambda_j(\overline{B})\} \le \|\overline{B}\|_\infty < 1$, $j = 1, 2, \ldots, n-1$. So, $0 < \lambda_j(I - \overline{B})^j = 1 - \lambda_j(\overline{B}) < 1$, $j = 1, 2, \ldots, n-1$. This implies that the spectral radius of the iteration matrix $I - \overline{B}$ is less than 1, i.e., $\rho(I - \overline{B}) < 1$. Therefore, the iterative format (14) converges. The proof is thus complete.

Theorem 1 tells us that the iteration process (14) always converges and we can obtain a satisfying approximate interpolant without solving a system of linear equations. However, as stated in [11], the collocation matrix resulting from the Bernstein basis is usually ill-conditioned, and the convergence rate of PIA is very slow. Therefore, it is necessary to introduce a preconditioner to speed up the convergence rate of PIA. In the following section, we propose a preconditioning technique for PIA by exploiting the properties of collocation matrix $\overline{B}$. □

3. Preconditioning Technique

3.1. Construction of Preconditioner. To derive a preconditioner for PIA, we first split $\overline{B}$ as $\overline{B} = \overline{B}_q + R$, where

$$R = \begin{pmatrix} 0 & \cdots & 0 & B_{u+q+1}^m(t_1) & \cdots & B_{m-v-1}^m(t_1) \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & B_{m-v-1}^m(t_{m-q-v-2}) \\ B_{u+1}^m(t_{u+q+1}) & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ B_{u+1}^m(t_{n-1}) & \cdots & B_{m-q-2r-2}^m(t_{n-1}) & 0 & \cdots & 0 \end{pmatrix}, \tag{18}$$

and $\overline{B}_q = \overline{B} - R$ is a band matrix with bandwidth $2q + 1$ $(0 \le q \le n-2)$.

Then, we define a diagonal matrix

$$D = \operatorname{diag}(d_1, d_2, \ldots, d_{n-1}), \tag{19}$$

such that $D\mathbf{e} = R\mathbf{e}$ with $\mathbf{e} = (1, 1, \ldots, 1)^T$.

Finally, let

$$M_q = \overline{B}_q + D \tag{20}$$

be the diagonally compensated reduction matrix of $\overline{B}$. Then, the matrix R is called the reduced matrix and D is called the compensation matrix for R [10, 12].

If M_q is invertible, it can serve as a preconditioner for (16) and yield the preconditioned system

$$M_q^{-1}\overline{B}\mathbf{X} = M_q^{-1}\overline{P}. \tag{21}$$

3.2. Preconditioned PIA. By applying Richardson method to the preconditioned system (21), we obtain

$$\begin{aligned} \mathbf{Q}^k &= \left(I - M_q^{-1}\overline{B}\right)\mathbf{Q}^{k-1} + M_q^{-1}\overline{P}, \\ &= \mathbf{Q}^{k-1} + M_q^{-1}\left(\overline{P} - \overline{B}\mathbf{Q}^{k-1}\right), \\ &= \mathbf{Q}^{k-1} + M_q^{-1}\boldsymbol{\Delta}^{k-1}, \quad k = 1, 2, \ldots, \end{aligned} \tag{22}$$

We refer the iterative process (22) as PPIA.

3.3. Convergence Analysis of PPIA

Theorem 2. *The preconditioner M_q defined in (20) is invertible.*

Proof. In the proof of Theorem 1, we show that $\overline{B}$ is positive stable. By Lemma 1, for any nontrivial $\mathbf{y}$, there exists a positive definite matrix K, such that $\operatorname{Re}[\mathbf{y}^* K\overline{B}\mathbf{y}] > 0$. According to the definitions of D and R, both the diagonal entries of $D - R$ and the sum of the absolute values of the nondiagonal entries in the j^{th} row of $D - R$ are d_j $(j \in \{1, \ldots, n-1\})$. Let $\lambda = a + b \cdot \mathbf{i} \in \mathbb{C}$ be any eigenvalue of $D - R$ and $\mathbf{x}$ be the eigenvector corresponding to λ. Then, it follows from Lemma 2 that $|\lambda - d_j| = \sqrt{(a - d_j)^2 + b^2} \le d_j$. Therefore, $0 \le a \le 2d_j$. This means that all the eigenvalues of $D - R$ have nonnegative real parts and $D - R$ is positive semistable. Hence, for the positive definite matrix K and the nontrivial $\mathbf{x}$, we have $\operatorname{Re}[\mathbf{x}^* K(D - R)\mathbf{x}] = \mathbf{x}^* K\mathbf{x}(\operatorname{Re}\lambda) \ge 0$.

From (20), we have $M_q = \overline{B} + D - R$, then $\operatorname{Re}[\mathbf{x}^* K M_q\mathbf{x}] = \operatorname{Re}[\mathbf{x}^* K(\overline{B} + D - R)\mathbf{x}] = \operatorname{Re}[\mathbf{x}^* K\overline{B}\mathbf{x} + \mathbf{x}^* K(D - R)\mathbf{x}] = \operatorname{Re}[\mathbf{x}^* K\overline{B}\mathbf{x}] + \operatorname{Re}[\mathbf{x}^* K(D - R)\mathbf{x}] = \operatorname{Re}[\mathbf{x}^* K\overline{B}\mathbf{x}] + \operatorname{Re}[\mathbf{x}^* K(\lambda\mathbf{x})] > 0$.

It follows from Lemma 1 that M_q is positive stable and there is no eigenvalue equal to zero. This completes the proof.

As it is stated in [13], the error matrix $\overline{E} = M_q - \overline{B}$ or the residual matrix $\overline{R} = I - M_q^{-1}\overline{B}$ is often used to measure the efficiency of preconditioner M_q.

Since $M_q = \overline{B} + D - R$, then

$$\|\overline{E}\|_\infty = \left\|M_q - \overline{B}\right\|_\infty = \|D - R\|_\infty = 2\|R\|_\infty. \tag{23}$$

From (18) and (19), we have

$$\|D\|_\infty = \|R\|_\infty = \max_{1\le j\le n-1} \sum_{\substack{u+1\le i\le m-v-1 \\ |i-j|>q}} B_i^m(t_j). \tag{24}$$

As is stated in [12], $B_{j+q+1}^m(t_j) > \cdots > B_n^m(t_j) = 0$, $B_{j-q-1}^m(t_j) > \cdots > B_0^m(t_j) = 0$. Thus, the sum of the entries of each column outside of the q diagonals decays to 0. This means that $\|\overline{E}\|_\infty$ approaches to 0 as q increases. Combined with (23), we conclude that the matrix M_q is a good approximation to $\overline{B}$ for large q. Thus, the spectrum of $M_q^{-1}\overline{B}$ is clustered around 1 for large q. □

Remark 2. As it is known to all, the iterative process (22) is convergent if the spectral radius of the iteration matrix is less than 1, that is, $\rho(I - M_q^{-1}\overline{B}) < 1$. Note that the eigenvalues of $I - M_q^{-1}\overline{B}$ are $1 - \lambda_i(M_q^{-1}\overline{B})$. In particular, when the eigenvalues of $M_q^{-1}\overline{B}$ are distributed in $[0, 1]$, we have $\rho(I - M_q^{-1}\overline{B}) = 1 - \lambda_{\min}(M_q^{-1}\overline{B})$. Therefore, the convergence of (22) mainly depend on the value of $\lambda_{\min}(M_q^{-1}\overline{B})$. Moreover, the larger the value of $\lambda_{\min}(M_q^{-1}\overline{B})$ is, the faster the PPIA converges. It would be great if $\lambda(M_q^{-1}\overline{B}) \in [0, 1]$. Unfortunately, we cannot give a rigid theoretical proof that $\lambda(M_q^{-1}\overline{B}) \in [0, 1]$. Numerous experiments indicate that the eigenvalues of $M_q^{-1}\overline{B}$ are separated in $[0, 1]$, and most of them are clustered around 0. Based on the above analysis, we make a conjecture that the eigenvalues of $M_q^{-1}\overline{B}$ are separated in $[0, 1]$.

Recalling that $M_q = \overline{B} + D - R$, we have $\overline{B}^{-1}M_q = \overline{B}^{-1}(\overline{B} + D - R) = I + \overline{B}^{-1}(D - R)$; hence,

$$\begin{aligned}\rho\left(\overline{B}^{-1}M_q\right) &= \rho\left[I + \overline{B}^{-1}(D - R)\right] \\ &\le 1 + \rho\left[\overline{B}^{-1}(D - R)\right] \le 1 + \left\|\overline{B}^{-1}\right\|\|D - R\|_\infty.\end{aligned} \tag{25}$$

Note that when $\lambda_i(M_q^{-1}\overline{B}) > 0$, we have

$$\lambda_{\min}\left(M_q^{-1}\overline{B}\right) = \frac{1}{\rho\left(\overline{B}^{-1}M_q\right)}. \tag{26}$$

According to (25) and the conjecture in Remark 2, we have

$$\begin{aligned}\rho\left(I - M_q^{-1}\overline{B}\right) &= 1 - \lambda_{\min}\left(M_q^{-1}\overline{B}\right) \\ &= 1 - \frac{1}{\rho\left(\overline{B}^{-1}M_q\right)} \le 1 - \frac{1}{1 + \left\|\overline{B}^{-1}\right\|\|D - R\|_\infty} < 1.\end{aligned} \tag{27}$$

Thus, PPIA converges.

Remark 3. In (27), we give a upper bound for $\rho(I - M_q^{-1}\overline{B})$. As mentioned earlier, $\|D - R\|_\infty$ tends to 0 as q increases; therefore, the upper bound $1 - (1/(1 + \|\overline{B}^{-1}\|\|D - R\|_\infty))$ also tends to 0. In particular, $\rho(I - M_q^{-1}\overline{B})$ equals to 0 when $q = n - 2$. Hence, the spectral radius $\rho(I - M_q^{-1}\overline{B})$ may decrease as q increases, i.e., the bigger the value of q is, the faster the PPIA format converges.

On the other hand, it should be pointed out that at the $k(k = 0, 1, \ldots,)^{\text{th}}$ iteration of PPIA, we have to calculate $M_q^{-1}\Delta^{(k)}$ or solve the equation $M_q\mathbf{X}^{(k)} = \Delta^{(k)}$ equivalently. It is costly, especially for large q. Therefore, we need to seek a tradeoff between the convergence rate and computational complexity. Experimentally, we found that $q \approx (n/2)$ is a suitable choice.

4. Applications of PPIA with Constraints at Endpoints

Polynomial approximation for higher order or rational Bézier curves has been a continuous hotspot in computer-aided geometric design (CAGD). A number of approximation methods have been proposed in the literature, for example, polynomial approximation for rational Bézier curves [14–20] and polynomial approximation for higher Bézier curves (degree reduction) [3, 21–28].

As a sample-based polynomial approximate method, Lu employed the weighted PIA (WPIA) to iteratively approximate the points sampled from the rational Bézier curves and achieved good approximations [17]. As mentioned in [19], Lu's method is slow due to the slow convergence of WPIA and cannot deal with polynomial approximation with higher order continuity conditions at the endpoints. In fact, polynomial approximations with constraints are more practical than those without constraints. Due to the effectiveness in interpolation, these problems resulted from Lu's method can be solved efficiently by employing the presented PPIA with constraints.

4.1. Polynomial Iterative Approximation for Higher Order and Rational Bézier Curves with Constraints. Given a rational Bézier curve of degree N

$$\mathbf{R}(t) = \frac{\sum_{j=0}^N \omega_j \mathbf{v}_j B_j^N(t)}{\sum_{j=0}^m \omega_j B_j^N(t)}, \tag{28}$$

where $\mathbf{v}_j$ and $\omega_j \in \mathbb{R}^+$ are the control points and the associated weights, respectively.

First, we sample $n + 1$ points $\{\mathbf{R}(t_j)\}_{j=0}^n$ at the parameter values $t_j\,(j = 0, 1, \ldots, n)$ and the derivatives $\{\mathbf{R}^{(r)}(t_0)\}_{r=1}^u$ and $\{\mathbf{R}^{(s)}(t_1)\}_{s=1}^v$ at endpoints, where $0 = t_0 < t_1 < \cdots < t_n = 1$. Then, PPIA for Bézier curves with constraints is employed to interpolate the points $\{\mathbf{R}(t_j)\}_{j=0}^n$ and the derivatives $\{\mathbf{R}^{(r)}(t_0)\}_{r=1}^u$ and $\{\mathbf{R}^{(s)}(t_1)\}_{s=1}^v$. Therefore, we can generate a sequence of m degree Bézier curves $\mathbf{C}^k(t)$, which are the polynomial approximations of $\mathbf{R}(t)$.

Remark 4. We remark here that if all the weights ω_j in (18) equal to 1, the rational Bézier curve (28) will degenerate into the Bézier curve of degree N. At this time, if $m < N$, it is the problem of approximating a given curve by a lower degree

Input: original curve $\mathbf{R}(t)$, largest admissible number $k_{\max}$, and outer stopping tolerance θ in (19)
Output: control points of the approximate curve, number of iterations k, and approximation error $L_p^{(k+1)}$
(1) Sample the points $\{\mathbf{R}(t_j)\}_{j=0}^{n}$ and the derivatives $\{\mathbf{R}^{(r)}(t_0)\}_{r=1}^{u}, \{\mathbf{R}^{(s)}(t_1)\}_{s=1}^{v}$
(2) Compute the initial error $L_2^{(0)}$ according to (31)
(3) For $k = 0, 1, \ldots, k_{\max}$
  (i) Employ PPIA for Bézier curves with constraints to generate the $(k+1)^{\text{th}}$ approximation curve
  (ii) Compute $L_2^{(k+1)}$ according to (31)
  (iii) If $L_2^{(k+1)} \geq \theta L_2^{(k)}$, break for
End for

Algorithm 1: Polynomial approximation for higher order and rational Bézier curves with constraints.

Bézier curve, which is an important approximation problem in CAGD, i.e., degree reduction of Bézier curves.

As stated in [17], the iterative method guarantees the convergence of data interpolation but does not necessary converge to the given curve $\mathbf{R}(t)$. Therefore, it is not necessary for PPIA to iterate infinitely. To ensure the computing efficiency, the iteration can be terminated if

$$L_p^{(k+1)} \geq \theta L_p^{(k)}, \quad \theta \in (0,1), \tag{29}$$

is satisfied, where $L_p^{(k)}$ is the approximation error at the k^{th} iteration given by

$$L_p^{(k)} = \left(\int_0^1 \left\| \mathbf{C}^k(t) - \mathbf{R}(t) \right\|^p \mathrm{d}t \right)^{(1/p)}, \quad 1 \leq p < \infty. \tag{30}$$

Since there may exist rational functions in the integrand of (30), we need an effective and stable numerical method to calculate the integral (30). We note in [29] that the Gaussian rule or/and the adaptive quadrature method would be a better choice for rational cases. Therefore, we employ the τ-order Gauss–Legendre rule to calculate the integral (30).

By solving the roots of the Legendre polynomial of degree τ, we can obtain Gauss points $x_i, i = 1, 2, \ldots,$ and then calculate τ positive weights ω_i. Finally, the integral in (30) can be evaluated approximately by

$$L_p^{(k)} \approx \left(\sum_{1 \leq i \leq \tau} \omega_i \left\| \mathbf{C}^k(x_i) - \mathbf{R}(x_i) \right\|^p \right)^{(1/p)}. \tag{31}$$

We refer the reader to read [30] for more details about the τ-order Gauss–Legendre quadrature. In our tests, we used 15-order Gauss–Legendre quadrature to estimate the approximation error $L_2^{(k)}$, i.e., $\tau = 15$ and $p = 2$.

Finally, we summarize the approximation algorithm in the following Algorithm 1.

5. Numerical Examples

In this section, several numerical experiments are given to illustrate the effectiveness of the proposed methods. All the numerical experiments were carried out on a computer with Intel(R) Core(TM) i5-5200U CPU @2.20 GHz by Matlab R2012b.

5.1. Tests of PIA and PPIA with Constraints. Since we have shown in Theorem 1 that the approximate interpolation curves $\mathbf{C}^k(t)$ interpolate the endpoints and the derivatives at the endpoints exactly, we need to measure the interpolation error at the points $\mathbf{p}_i, i = 1, 2, \ldots, n-1$. Therefore, the interpolation error of the k^{th} approximate interpolation curve $\mathbf{C}^k(t)$ can be expressed by

$$\varepsilon^{(k)} = \max_{1 \leq i \leq n-1} \left\| \mathbf{C}^k(t_i) - \mathbf{p}_i \right\|, \tag{32}$$

where the norm is the Euclidean norm.

Example 1. We sample 9 points $\mathbf{p}_i (i = 0, 1, \ldots, 8)$, $r\,(r = 1, 2)^{\text{th}}$ derivatives $\mathbf{d}_0^r$, and $s\,(s = 1, 2)^{\text{th}}$ derivatives $\mathbf{d}_n^s$ from the semicircle of radius 5 in the following way:

$$\begin{aligned} &\mathbf{p}_i = \left(5\cos\left(\frac{\pi}{8}i\right), 5\sin\left(\frac{\pi}{8}i\right) \right), \quad i = 0, 1, \ldots, 8; \\ &\mathbf{d}_0^1 = (0, 5), \mathbf{d}_8^1 = (0, -5); \\ &\mathbf{d}_0^2 = (-5, 0), \mathbf{d}_8^2 = (5, 0). \end{aligned} \tag{33}$$

Example 2. We sample 19 points $\mathbf{p}_i (i = 0, 1, \ldots, 18)$, $r\,(r = 1, 2, 3)^{\text{th}}$ derivatives $\mathbf{d}_0^r$, and $s\,(s = 1, 2, 3)^{\text{th}}$ derivatives $\mathbf{d}_n^s$ from the helix of radius 5 in the following way:

$$\begin{aligned} &\mathbf{p}_i = \left(5\cos\left(\frac{\pi}{3}i\right), 5\sin\left(\frac{\pi}{3}i\right), \frac{\pi}{3}i \right), \quad i = 0, 1, \ldots, 18; \\ &\mathbf{d}_0^1 = \mathbf{d}_{18}^1 = (0, 5, 1); \\ &\mathbf{d}_0^2 = \mathbf{d}_{18}^2 = (-5, 0, 0). \end{aligned} \tag{34}$$

PIA and PPIA formats for constraints at the endpoints are employed to test Example 1 and Example 2. When we test PPIA, we take the half bandwidth $q = 4$ in Example 1 and $q = 10$ in Example 2. In Table 1, we list the spectral radii of the iteration matrices of PIA and PPIA with different constraints at the endpoints. For one thing, the spectral radii of the iteration matrices of PIA and PPIA are less than 1; this means that both PIA and PPIA formats converge. For another, the spectral radii of PPIA are much less that those of PIA; hence, we can expect that PPIA would have a better convergence behavior than PIA.

We list in Table 2 the interpolation errors of PIA and PPIA when we test Example 1, and we list in Table 3 the

Table 1: Spectral radii of iteration matrices of PIA and PPIA in Example 1 and Example 2.

$\mathbf{d}_0^u$ and $\mathbf{d}_n^v$	Example 1		Example 2	
	PIA	PPIA	PIA	PPIA
$u = v = 1$	0.99918298	0.23846897	0.99999985	0.14929065
$u = 1, v = 2$	0.99785411	0.05994428	0.99999985	0.32779082
$u = 2, v = 1$	0.99785411	0.05994428	0.99999985	0.32779082
$u = v = 2$	0.99964744	0.57887371	0.99999985	0.24570266

Table 2: Interpolation errors of PIA and PPIA in Example 1 with different constraints at endpoints.

k	$u = v = 1$		$u = 1, v = 2$		$u = 2, v = 1$		$u = v = 2$	
	PIA	PPIA	PIA	PPIA	PIA	PPIA	PIA	PPIA
1	5.4395$e-01$	3.9638$e-03$	1.4104e00	4.3811$e-02$	1.4104e00	4.3811$e-02$	9.1860$e-01$	1.8233$e-02$
2	3.8274$e-01$	1.4605$e-04$	9.1358$e-01$	2.9530$e-03$	9.1358$e-01$	2.9530$e-03$	7.7540$e-01$	7.6179$e-04$
3	2.8078$e-01$	5.3842$e-06$	7.7453$e-01$	1.7897$e-04$	7.7453$e-01$	1.7897$e-04$	6.6697$e-01$	3.1829$e-05$
4	2.1528$e-01$	1.9849$e-07$	6.6933$e-01$	1.0740$e-05$	6.6933$e-01$	1.0740$e-05$	5.8133$e-01$	1.3299$e-06$
5	1.7055$e-01$	7.3177$e-09$	5.8848$e-01$	6.4387$e-07$	5.8848$e-01$	6.4387$e-07$	5.1179$e-01$	5.5567$e-08$
10	1.1297$e-01$	5.6300$e-14$	3.6339$e-01$	5.2694$e-13$	3.6339$e-01$	4.9105$e-13$	3.4893$e-e-01$	8.0428$e-15$
15	8.6368$e-02$	5.5417$e-e-14$	3.4564$e-01$	8.9706$e-14$	3.4564$e-01$	2.4995$e-14$	3.4688$e-01$	8.0428$e-15$
20	6.4608$e-02$	5.5417$e-14$	3.2913$e-01$	8.9706$e-14$	3.2913$e-01$	2.4995$e-14$	3.0648$e-01$	8.0428$e-15$
40	2.4501$e-02$	5.5417$e-14$	2.4740$e-01$	8.9706$e-14$	2.4740$e-01$	2.4995$e-14$	1.6225$e-01$	8.0428$e-15$
50	2.2440$e-02$	5.5417$e-14$	2.1753$e-01$	8.9706$e-14$	2.1753$e-01$	2.4995$e-14$	1.2254$e-01$	8.0428$e-15$

Table 3: Interpolation errors of PIA and PPIA in Example 2 with different constraints at endpoints.

k	$u = v = 1$		$u = 1, v = 2$		$u = 2, v = 1$		$u = v = 2$	
	PIA	PPIA	PIA	PPIA	PIA	PPIA	PIA	PPIA
1	4.6885e00	7.5971$e-05$	1.1145e01	5.9422$e-03$	1.1191e01	5.9448$e-03$	9.4468e00	2.1520$e-03$
2	4.2701e00	2.6533$e-06$	7.0081e00	2.3159$e-03$	6.9937e00	2.3164$e-03$	5.7317e00	1.9501$e-04$
3	3.7030e00	1.5379$e-07$	5.8458e00	7.6479$e-04$	5.8565e00	7.6496$e-04$	4.9157e00	3.7394$e-05$
4	3.1942e00	1.9400$e-08$	4.6120e00	2.5081$e-04$	4.6235e00	2.5087$e-04$	3.5536e00	9.0175$e-06$
5	2.7764e00	2.8613$e-09$	3.5412e00	8.2217$e-05$	3.5411e00	8.2235$e-05$	2.4739e00	2.2132$e-06$
10	1.5062e00	2.1517$e-13$	2.4607e00	3.1113$e-07$	2.4604e00	3.1120$e-07$	1.9172e00	1.9817$e-09$
20	6.7818$e-01$	1.7061$e-14$	1.6841e00	3.3866$e-12$	1.6823e00	4.6786$e-12$	1.3289e00	1.9304$e-13$
50	2.5157$e-01$	1.3557$e-14$	1.2789e00	9.5874$e-13$	1.2786e00	8.1082$e-13$	9.9004$e-01$	2.2176$e-13$
100	9.7764$e-02$	2.4470$e-14$	7.6721$e-01$	9.6635$e-13$	7.6715$e-01$	9.3884$e-13$	6.2363$e-01$	1.6799$e-13$

interpolation errors of PIA and PPIA when we test Example 2. As we can see from Tables 2 and 3, the interpolation errors of PIA and WPIA decrease as the number of iterations increases, which indicates that both PIA and PPIA converge. With the same iterations, the interpolate errors of PPIA are much less than those of PIA. This means that the preconditioning technique can accelerate the convergence rate of PIA significantly.

In Figures 1 and 2, we display the interpolation Bézier curves (dashed lines) when approximating the points with different derivatives given in Example 1. We also show in Figures 1 and 2 the semicircle (solid lines) for comparison. It should be pointed out that the approximate interpolation Bézier curves generated by PPIA almost coincide with the semicircle, no matter which endpoint conditions are used. In Figures 3 and 4, we display the interpolation Bézier curves when approximating the points combined with different derivatives given in Example 2. All the approximate interpolation Bézier curves in Figures 1–4 are obtained by preforming 5 PIA or PPIA iterations. Clearly, the IPPIA can yield better approximations than PIA.

5.2. Tests of Polynomial Approximation for Higher Order and Rational Bézier Curves. First, we employ Algorithm 1 to test the degree reduction of higher order Bézier curves in Example 3 and then test the polynomial approximation for rational Bézier curve in Example 4 and Example 5. In our tests, we take the endpoint interpolation conditions ($u = v = 1$) and take the half bandwidth $q = \lceil n/2 \rceil$ when we construct the preconditioner for PIA, where $\lceil \cdot \rceil$ is the round-up operation. When we test Example 3 and Example 4, we take $\theta = 0.9$. Again, when we test Example 5, we take $\theta = 0.98$.

Example 3 (Degree reduction [17]). A Bézier curve of degree 15 is defined by the control points $(0, 0)$, $(1.5, -2)$, $(4.5, -1)$, $(9, 0)$, $(4.5, 1.5)$, $(2.5, 3)$, $(0, 5)$, $(-4, 8.5)$, $(3, 9.5)$, $(4.4, 10.5)$, $(6, 12)$, $(8, 11)$, $(9, 10)$, $(9.5, 5)$, $(7, 6)$, and $(5, 7)$.

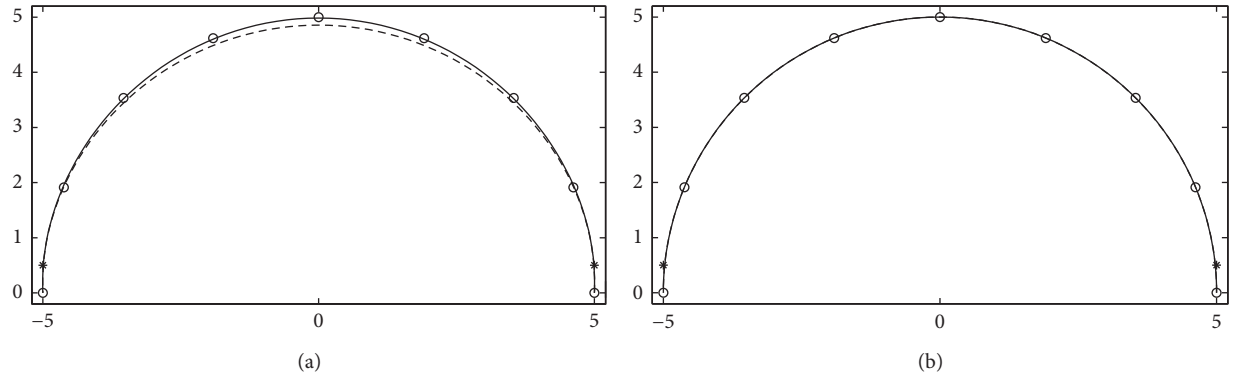

Figure 1: Bézier curves interpolating $\mathbf{d}_0^1$ and $\mathbf{d}_8^1$ in Example 1. (a) PIA. (b) PPIA.

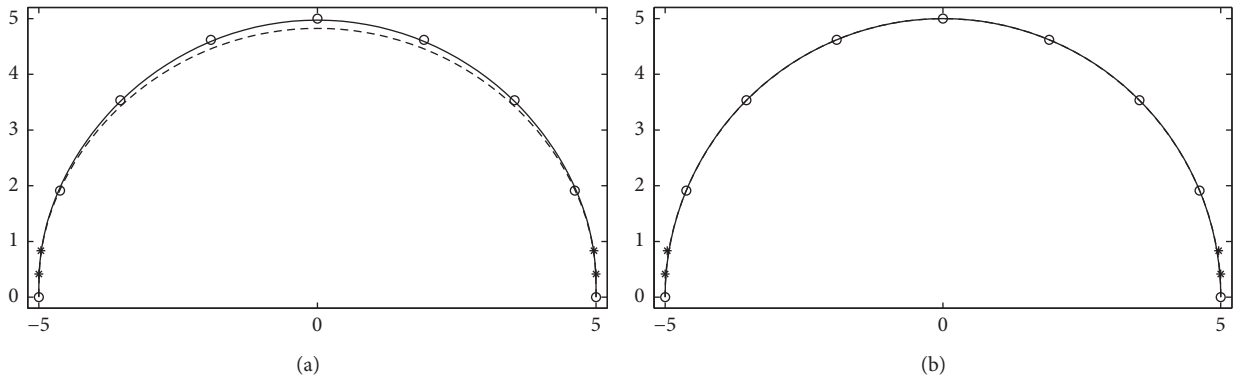

Figure 2: Bézier curves interpolating $\mathbf{d}_0^2$ and $\mathbf{d}_8^2$ in Example 1. (a) PIA. (b) PPIA.

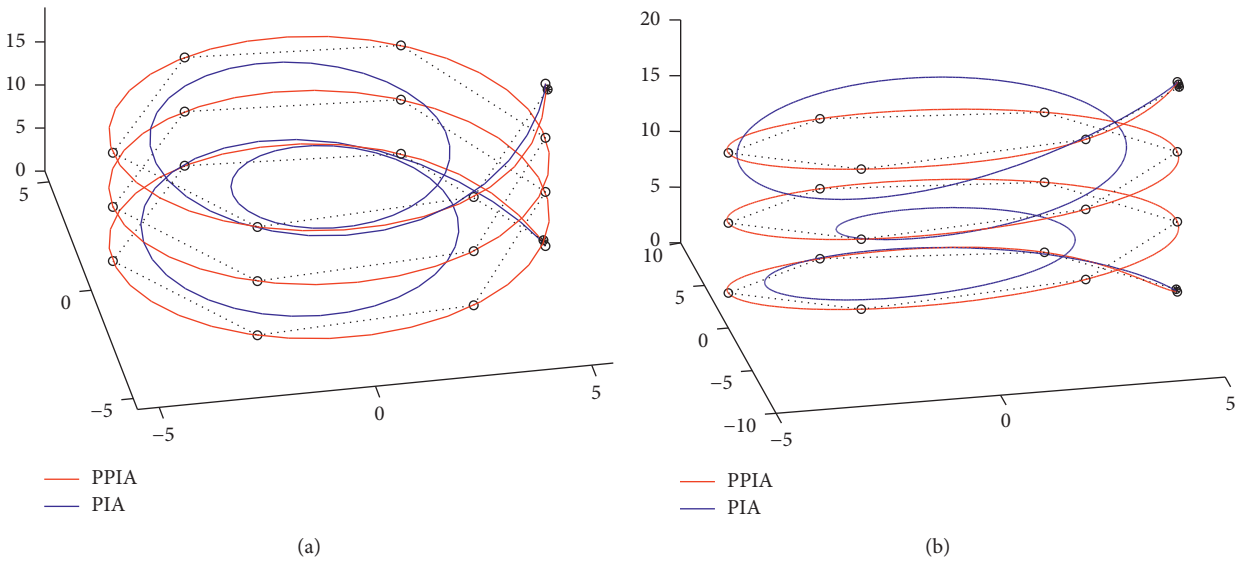

Figure 3: Bézier curves obtained by performing 5 PIA or PPIA iterations in Example 2. (a) Interpolating d_0^1 and d_{18}^1. (b) Interpolating d_0^1 and d_{18}^2.

By employing Algorithm 1, we use the Bézier curves of degree $m\,(m = 6, 7, \ldots, 14)$ to approximate the Bézier curve given in Example 3. In Table 4, we list the iteration number k, the approximation errors $L_2^{(k)}$, and the elapsed CPU time T (in seconds) of Algorithm 1; the numerical results run by Lu's method ([17]) are also listed for comparison. The results for degree reduction with constraints, reported in Table 4, indicate that Algorithm 1 outperforms Lu's method, in terms of the approximation error and the computing time.

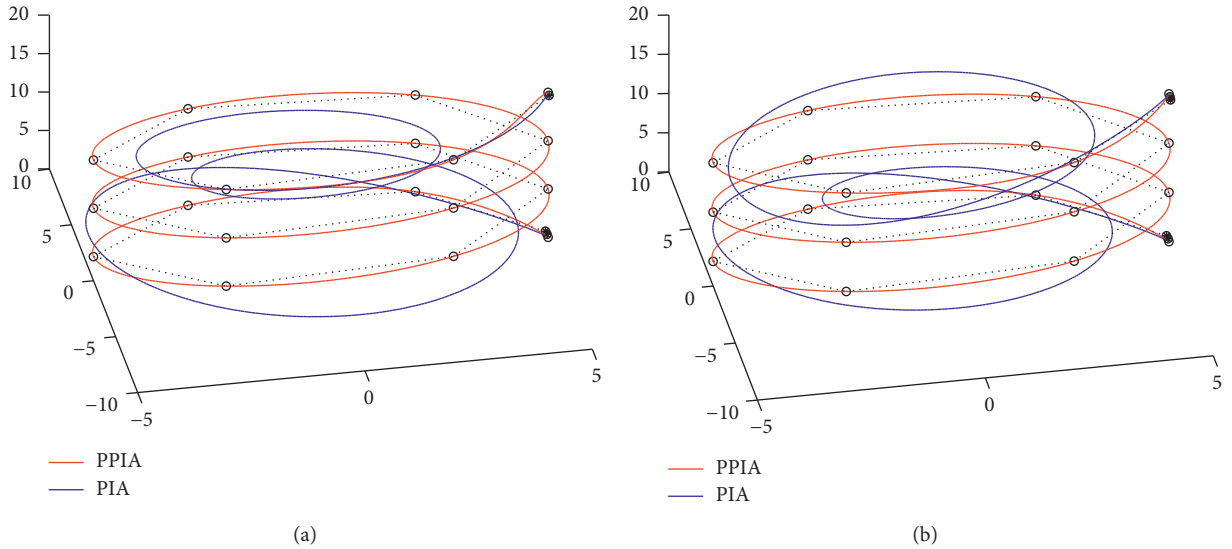

Figure 4: Bézier curves obtained by performing 5 PIA or PPIA iterations in Example 2. (a) Interpolating d_0^2 and d_{18}^1. (b) Interpolating d_0^2 and d_{18}^2.

Table 4: Numerical results of Algorithm 1 and Lu's method in Example 3.

m	Algorithm 1			Lu's method		
	k	$L_2^{(k)}$	T(s)	k	$L_2^{(k)}$	T(s)
6	1	$3.6163e-01$	0.59	3	$4.9761e-01$	2.14
7	1	$1.3234e-01$	0.60	4	$3.6118e-01$	2.59
8	1	$9.2304e-02$	0.60	7	$1.9136e-01$	3.58
9	1	$2.1803e-02$	0.60	9	$1.0012e-01$	3.99
10	1	$8.6645e-03$	0.60	9	$7.4032e-02$	4.14
11	1	$5.8744e-03$	0.61	9	$6.3833e-02$	4.07
12	2	$2.5835e-03$	0.94	11	$4.7558e-02$	4.67
13	1	$2.3246e-04$	0.61	11	$4.5005e-02$	4.79
14	3	$1.1873e-04$	1.23	15	$2.6818e-02$	6.09

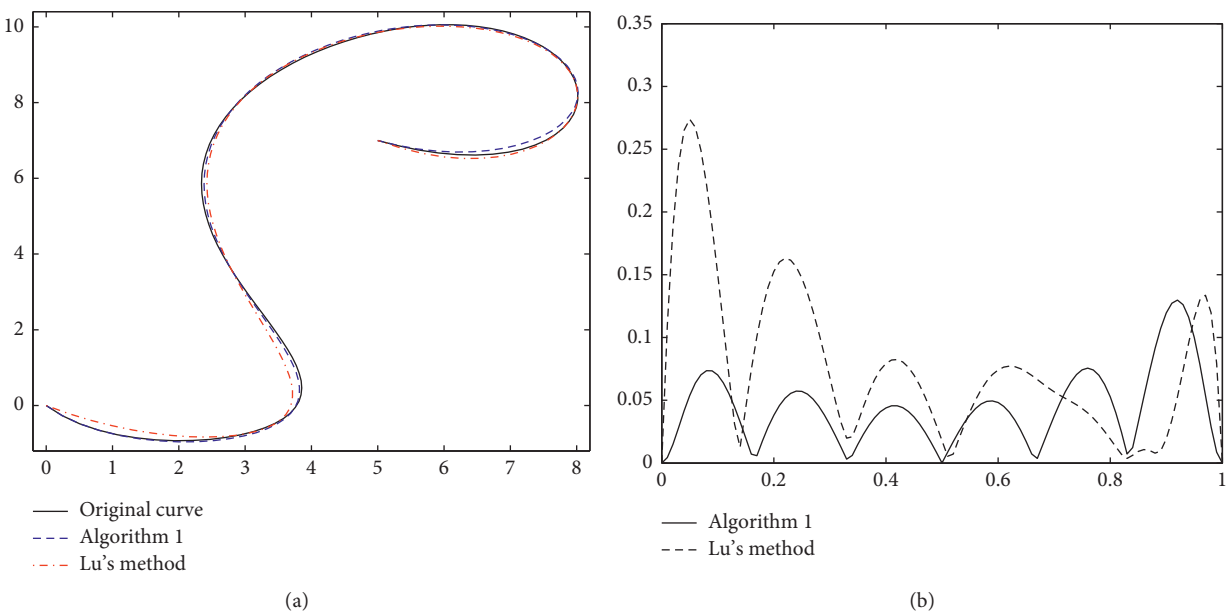

Figure 5: Reduction from degree 15 to 8 in Example 3. (a) Comparison of polynomial approximations. (b) The corresponding error distance curves.

Table 5: Numerical results of Algorithm 1 and Lu's method in Examples 4 and 5.

Example 4							Example 5						
m	Algorithm 1			Lu's method			m	Algorithm 1			Lu's method		
	k	$L_2^{(k)}$	T(s)	k	$L_2^{(k)}$	T(s)		k	$L_2^{(k)}$	T(s)	k	$L_2^{(k)}$	T(s)
8	1	$1.9356e+00$	0.58	5	$1.9556e+00$	1.76	9	1	$5.7514e-01$	0.59	8	$9.9194e-01$	2.64
9	1	$1.0215e+00$	0.59	5	$1.6452e+00$	1.75	10	2	$2.9092e-01$	0.87	10	$7.6682e-01$	3.30
10	1	$6.6233e-01$	0.58	5	$1.4297e+00$	1.80	11	1	$1.5379e-01$	0.59	12	$5.8658e-01$	3.96
11	1	$4.0403e-01$	0.60	5	$1.2715e+00$	1.80	12	2	$6.9603e-02$	0.92	16	$4.2321e-01$	5.23
12	1	$2.6760e-01$	0.59	5	$1.1512e+00$	1.78	13	1	$4.3342e-02$	0.63	18	$3.1705e-01$	5.75
13	1	$1.6397e-01$	0.59	5	$1.0575e+00$	1.79	14	3	$2.0350e-02$	1.19	18	$2.4794e-01$	5.86
14	1	$1.2175e-01$	0.58	7	$7.7411e-01$	2.39	15	2	$1.5264e-02$	0.89	20	$1.8593e-01$	6.47

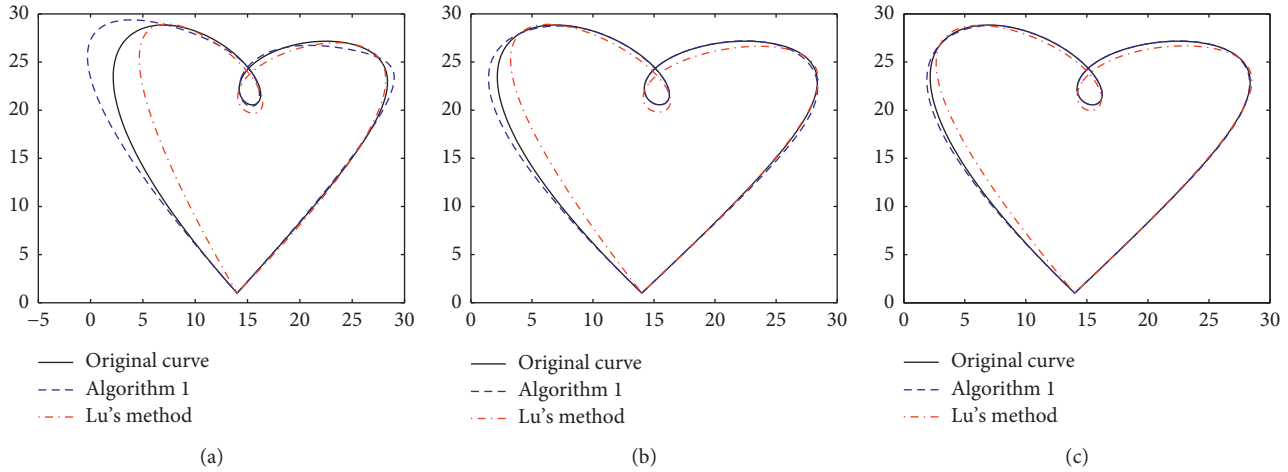

Figure 6: Polynomial approximations of the rational Bézier curve in Example 4 with different m. (a) $m = 9$. (b) $m = 12$. (c) $m = 14$.

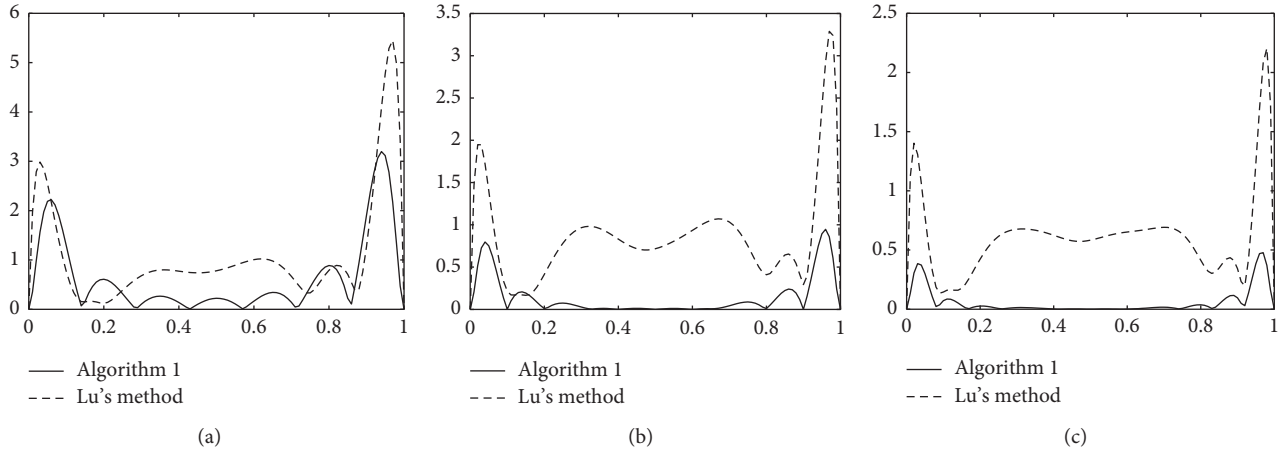

Figure 7: The corresponding error distance curves of the approximations in Figure 6. (a) $m = 9$. (b) $m = 12$. (c) $m = 14$.

In Figure 5(a), we show the approximations of degree 8 obtained by Algorithm 1 (blue dashed line) and Lu's method (red dashdot line). In Figure 5(b), we display the corresponding error distance curves of the approximations shown in Figure 5(a). It is evident from Figure 5 that Algorithm 1 can produce better approximations than Lu's method. What is more, the approximations obtained by Algorithm 1 are C^1-continuity at two endpoints, whereas the approximations obtained by Lu's method only satisfy the simplest C^0-continuity.

Example 4. (Polynomial approximation for rational Bézier curves [19]). A rational Bézier curve of degree 8 is defined by the control points $(14, 1)$, $(34, 25)$, $(40, 38)$, $(-12, 24)$,

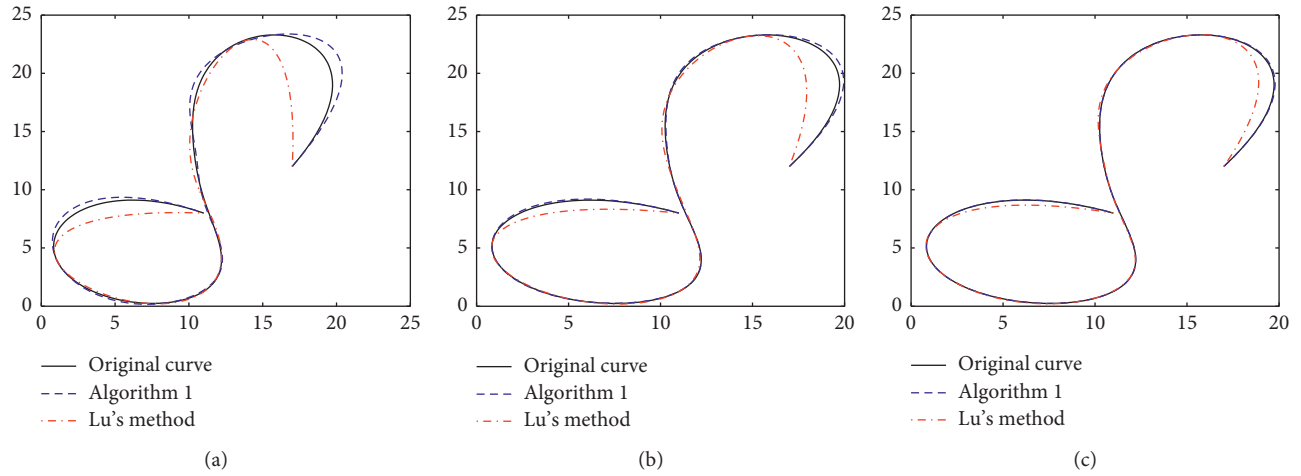

Figure 8: Polynomial approximations of the rational Bézier curve in Example 5 with different m. (a) $m = 10$. (b) $m = 12$. (c) $m = 15$.

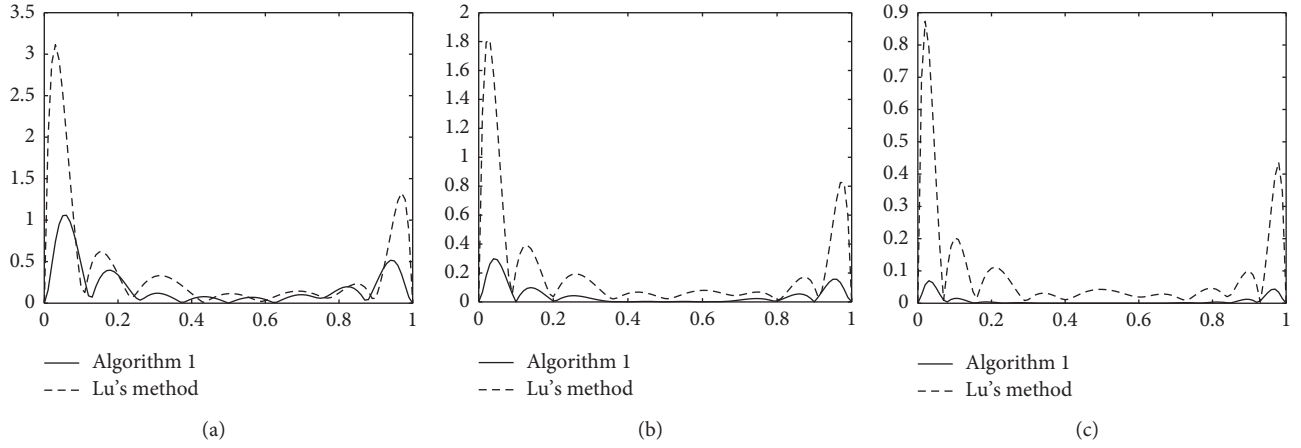

Figure 9: The corresponding error distance curves of the approximations in Figure 8. (a) $m = 10$. (b) $m = 12$. (c) $m = 15$.

$(5, 21)$, $(26, 7)$, $(18, 41)$, $(-13, 34)$, and $(14, 1)$ and the associated weights $1, 2, (1/3), 2, 2, (1/3), 2$, and 1.

Example 5. (Polynomial approximation for rational Bézier curves [19])**.** A rational Bézier curve of degree 9 is defined by the control points $(17, 12)$, $(32, 34)$, $(-23, 24)$, $(33, 6)$, $(-23, 15)$, $(25, 3)$, $(30, -2)$, $(-5, -8)$, $(-5, 15)$, $(11, 8)$, and the associated weights $1, 2, 3, 6, 4, 5, 3, 4, 2, 1$.

We employ Algorithm 1 and Lu's method to generate polynomial approximations for the rational Bézier curves given in Example 4 and Example 5. In Table 5, we list the iteration number k, the approximation error $L_2^{(k)}$, and the elapsed CPU time T (in seconds) of Algorithm 1 and Lu's method. The results indicate that Algorithm 1 also performs better than Lu's method for polynomial approximation for rational Bézier curves with constraints.

The $m\,(m = 9, 12, 15)$ degree approximations for the rational Bézier curve in Example 4 and their corresponding error distance curves are shown in Figures 6 and 7, respectively. Again, $m\,(m = 10, 12, 15)$ degree approximations for the rational Bézier curve in Example 5 and their corresponding error distance curves are shown in Figures 8 and 9, respectively. It is already clear from these figures the reliability of the proposed approximation algorithm.

6. Conclusion

The presented PIA format for Bézier curves iteratively interpolates not only a set of points but also derivatives at endpoints. It can construct higher order continuous piecewise interpolation Bézier curves without solving a linear system directly. To accelerate the convergence rate of PIA, the preconditioning technique is introduced. Numerical examples demonstrate that the given methods are effective, and the convergence rate of PIA is accelerated significantly by the presented preconditioning technique.

Furthermore, due to the efficient performance in data interpolation, the presented PPIA for Bézier curves with

constraints is applied to approximate higher order and rational Bézier curves. Our numerical experiments show that the proposed method achieves higher efficiency and results in a better approximation than the similar sample-based polynomial approximation method. More importantly, they can satisfy C^k $(k = 1, 2, \ldots,)$ continuity at the endpoints.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (11771453), Natural Science Foundation of Hunan Province (2020JJ5267), and Scientific Research Funds of Hunan Provincial Education Department (CX20201192 and 19B301).

References

[1] S.-i. Gofuku, S. Tamura, and T. Maekawa, "Point-tangent/point-normal B-spline curve interpolation by geometric algorithms," *Computer-Aided Design*, vol. 41, no. 6, pp. 412–422, 2009.

[2] S. Okaniwa, A. Nasri, H. Hongwei Lin, A. Abbas, Y. Kineri, and T. Maekawa, "Uniform B-spline curve interpolation with prescribed tangent and curvature vectors," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 9, pp. 1474–1487, 2012.

[3] Wozny, Pawel, Gospodarczyk et al., "Degree reduction of composite Bezier curves," *Applied Mathematics and Computation*, vol. 293, pp. 40–48, 2017.

[4] G. E. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, Morgan Kaufmann Publishers Inc, Burlington, MA, USA, 2002.

[5] A. M. Bica, "Optimizing at the end-points the Akima's interpolation method of smooth curve fitting," *Computer Aided Geometric Design*, vol. 31, pp. 245–257, 2014.

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

[7] R. T. Farouki, "The Bernstein polynomial basis: a centennial retrospective," *Computer Aided Geometric Design*, vol. 29, no. 6, pp. 379–419, 2012.

[8] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.

[9] H.-W. Lin, H.-J. Bao, and G.-J. Wang, "Totally positive bases and progressive iteration approximation," *Computers & Mathematics with Applications*, vol. 50, no. 3-4, pp. 575–586, 2005.

[10] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[11] A. Marco and J. Martínez, "A fast and accurate algorithm for solving Bernstein-Vandermonde linear systems," *Linear Algebra and Its Applications*, vol. 422, pp. 616–628, 2006.

[12] C. Liu and Z. Liu, "Progressive iterative approximation with preconditioners," *Mathematics*, vol. 8, no. 9, p. 1503, 2020.

[13] M. Benzi, "Preconditioning techniques for large linear systems: a survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[14] G.-J. Wang, T. W. Sederberg, and F. Chen, "On the convergence of polynomial approximation of rational functions," *Journal of Approximation Theory*, vol. 89, no. 3, pp. 267–288, 1997.

[15] M. S. Floater, "High order approximation of rational curves by polynomial curves," *Computer Aided Geometric Design*, vol. 23, no. 8, pp. 621–628, 2006.

[16] Y. Huang, H. Su, and H. Lin, "A simple method for approximating rational Bézier curve using Bézier curves," *Computer Aided Geometric Design*, vol. 25, no. 8, pp. 697–699, 2008.

[17] L. Lu, "Sample-based polynomial approximation of rational Bézier curves," *Journal of Computational and Applied Mathematics*, vol. 235, no. 6, pp. 1557–1563, 2011.

[18] H.-J. Cai and G.-J. Wang, "Constrained approximation of rational Bézier curves based on a matrix expression of its end points continuity condition," *Computer-Aided Design*, vol. 42, no. 6, pp. 495–504, 2010.

[19] S. Lewanowicz, P. Woźny, and P. Keller, "Polynomial approximation of rational Bézier curves with constraints," *Numerical Algorithms*, vol. 59, no. 4, pp. 607–622, 2012.

[20] Q. Hu and H. Xu, "Constrained polynomial approximation of rational Bézier curves using reparameterization," *Journal of Computational and Applied Mathematics*, vol. 249, pp. 133–143, 2013.

[21] M. Eck, "Degree reduction of Bézier curves," *Computer Aided Geometric Design*, vol. 10, no. 3-4, pp. 237–251, 1993.

[22] P. Bogacki, S. E. Weinstein, and Y. Xu, "Degree reduction of Bézier curves by uniform approximation with endpoint interpolation," *Computer-Aided Design*, vol. 27, no. 9, pp. 651–661, 1995.

[23] G.-D. Chen and G.-J. Wang, "Optimal multi-degree reduction of Bézier curves with constraints of endpoints continuity," *Computer Aided Geometric Design*, vol. 19, no. 6, pp. 365–377, 2002.

[24] H. Sunwoo, "Matrix representation for multi-degree reduction of Bézier curves," *Computer Aided Geometric Design*, vol. 22, no. 3, pp. 261–273, 2005.

[25] P. Wozny and S. Lewanowicz, "Multi-degree reduction of Bézier curves with constraints, using dual Bernstein basis polynomials," *Computer Aided Geometric Design*, vol. 26, pp. 566–579, 2009.

[26] L. Zhou, Y. Wei, and Y. Yao, "Optimal multi-degree reduction of Bézier curves with geometric constraints," *Computer-Aided Design*, vol. 49, pp. 18–27, 2014.

[27] L. Lu, "Some improvements on optimal multi-degree reduction of Bézier curves with geometric constraints," *Computer-Aided Design*, vol. 59, pp. 39–42, 2015.

[28] L. Lu and X. Xiang, "Note on multi-degree reduction of B8A6zier curves via modified Jacobi-Bernstein basis transformation," *Journal of Computational & Applied Mathematics*, vol. 315, pp. 65–69, 2016.

[29] W. Gautschi, "Gauss-type quadrature rules for rational functions," *Numerical Integration IV*, vol. 112, pp. 111–130, 1993.

[30] J. Mathews and F. DFink, *Numerical Methods Using MATLAB*, Pearson, New York, NY, USA, 2004.