Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 30(5): 1799–1812.
DOI: 10.3934/era.2022091
Received: 28 December 2021
Revised: 19 March 2022
Accepted: 24 March 2022
Published: 30 March 2022

Research article

Improved algorithms for determining the injectivity of 2D and 3D rational Bézier curves

Xuanyi Zhao[1], Jinggai Li[2], Ying Wang[1] and Chungang Zhu[3,*]

[1] School of Science, Dalian Maritime University, Dalian 116026, China
[2] College of Mathematics and Information Science, Henan Normal University, Xinxiang 453007, China
[3] School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China

* **Correspondence:** Email: cgzhu@dlut.edu.cn; Tel: +860411847083518116.

Abstract: Bézier curves and surfaces are important to computer-aided design applications. This paper presents algorithms for checking the injectivity of 2D and 3D Bézier curves. An injective Bézier curve or surface is one that has no self-intersections. The proposed algorithms use recently proposed sufficient and necessary conditions under which Bézier curves are guaranteed to be non-self-intersecting. As well as a rigorous derivation of the proposed algorithms, we present a series of examples and derive the complexity and computation times of the proposed algorithms. We find that the complexity our algorithms is approximately $O(m)$, representing an improvement over previous injectivity-checking algorithms.

Keywords: injectivity; Bézier curve; monotone chain; algorithm analysis

1. Introduction

Bézier techniques bring sophisticated mathematical concepts into highly geometric and intuitive forms, and are thus at the core of computer-aided geometric design and 3D modeling. An in-depth introduction to Bézier techniques can be found in many textbooks [1–9], and industrial applications of Bézier techniques have been widely described [10–13]. Bézier proposed the UNISURF system [10, 14, 15] for the design of curves and surfaces based on approximations in 1962. Forrest [16] and Gordon and Riesenfeld [17] revealed the relationship between Bézier curves and Bernstein polynomials, while Farin [18] gave a description of rational Bézier curves. Various interesting properties characterize Bézier curves [19], such as endpoint interpolation, symmetry, affine invariance, convex hull, and diminishing variation [3, 20–22]. The theories of progressive-iterative approximation (PIA), least-squares progressive iterative approximation (LSPIA) and extended progressive-iterative approximation (ExPIA) of Bézier curves can be found in literature [23]. These methods have widely applications in academic studies and engineering practices as shown in [24]. Some effective algorithms and applications of Bézier curves can be found in [16, 25–27].

Hoffmann [28] pointed out that the intersection problem was fundamental in both solid modeling and the integration of geometric objects. Self-intersecting curves or surfaces may lead to unexpected results in geometric modeling and image morphing. For example, in human-face deformation, the existence of a fold (self-intersection of a curve or surface) will lead to information loss. Numerous studies on the self-intersection problem have presented calculations and proofs of the existence of self-intersections. The curve–curve algorithm proposed by Lasser [29] calculates the self-intersections of Bézier curves by subdividing control polygons using de Casteljau's algorithm, and an algorithm for calculating the self-intersections of B-spline curves were proposed by Tiller et al. [30].

Craciun et al. [31] derived the sufficient and necessary conditions that guarantee the injectivity of toric-Bézier patches, and it is known that injective Bézier surfaces have no self-intersections. Their results are suitable for Bézier curves as they are the one-dimensional analogue of toric-Bézier patches. However, there is a minor flaw in their result, which was corrected for 2D patches by Sottile et al. [32]. For rational Bézier curves, Zhu et al. [33] proposed a sufficient and necessary condition on the control polygons that guarantees the curves have no self-intersections for any choice of positive weights. Zhu et al. [33] also proposed an algorithm for checking the injectivity of rational Bézier curves. They also explained that the complexity of their algorithm is $O(m^5)$ for spacial cases (and is $O(m^4)$ for 2D cases). Although this algorithm terminates quickly in the case of self-intersecting curves, a less complex algorithm is still necessary. Yu et al. [34] proposed an improved algorithm for checking the injectivity of 2D toric surface and applied their algorithm to iso-geometric analysis. Conditions for checking injectivity of toric volumes with arbitrary positive weights were proposed by Yu et al. [35] and geometric conditions of injective 3D Bézier volumes were proposed by Zhao et al. [36].

Using injectivity results of Bézier curves proposed by Craciun et al. [31], we derive a new, less complex algorithm for checking the injectivity of Bézier curves. It has been proved that rational Bézier curves have no self-intersections if and only if their control points and lattice points are strongly compatible. Using the theory of monotone chains [37], Preparata et al. [37] proposed an algorithm of linear complexity that tests for monotonicity. Using the definition of a monotone chain, we propose an equivalent theorem for the condition of the injectivity of Bézier curves. This reduces the complexity of algorithms for checking the injectivity of rational Bézier curves. In this paper, we present two algorithms for checking the injectivity of 2D and 3D rational Bézier curves that the complexity of them is approximately $O(m)$.

The rest of paper is organized as follows. Section 2 introduces some related work and improved algorithms are proposed. The complexity of the algorithms are also analyzed. Some examples are shown in Section 3. Finally, we conclude the paper and present the future work in Section 4.

2. Algorithms

Looking at the history of computer-aided geometric design in industry, it can be argued that rational techniques and representations are at the root of geometric modeling [19]. The definition of rational Bézier curves have been widely used. Krasauskas [38] defined toric surface patches as an extension of rational Bézier curves in one dimension, called toric-Bézier patches. To complete our theorem and proof, we present the following definition of toric Bézier curves.

Definition 1. *Consider lattice points $\mathcal{A} = \{0, 1, 2, \ldots, m\} \subset \mathbb{R}$, then its convex hull is $\Delta = [0, m]$. Given control points $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^d, | i = 0, 1, \ldots, m\}$ $(d \in \{2, 3\})$. A toric-Bézier curve (rational Bézier curve) of degree m is a parametric curve that is described by the control points $\mathbf{p}_i$, weights $\omega = \{\omega_i | i = 0, 1, \ldots, m\}$. The map $F_{\mathcal{A},\omega,\mathcal{P}} : \Delta \to \mathbb{R}^d$ is defined as:*

$$F_{\mathcal{A},\omega,\mathcal{P}}(x) = \frac{\sum\limits_{i=0}^{m} \omega_i \mathbf{p}_i \beta_i^m(x)}{\sum\limits_{i=0}^{m} \omega_i \beta_i^m(x)}, x \in \Delta. \tag{2.1}$$

The basis functions are defined as

$$\beta_i^m(x) = c_i (1 - x)^i x^{m-i}. \tag{2.2}$$

The image $F_{\mathcal{A},\omega,\mathcal{P}}$ is called toric Bézier curve. The functions $\beta_i^m(x)$ are called toric Bézier basis.

It is easy to check that these functions are non negative and have no common zeros on Δ. The polygon obtained by connecting adjacent control points is called the control polygon, and is denoted by P. When the coefficients $c_i = \binom{m}{i} m^{-m}$ and $x = mt$, they are the Bernstein basis functions:

$$B_i^m(t) = \binom{m}{i} (1 - t)^i t^{m-i}. \tag{2.3}$$

If the parametric domain is changed to $[0, 1]$, then the toric Bézier curve is right the rational Bézier curve. This means the definition of toric Bézier curve is equivalent to the definition of rational Bézier curve. For convenience of analysis, we use Definition 1.

In 2009, Craciun et al. [31] proposed several results on injectivity of toric surfaces. In this section, we illustrate the lower dimension results for Bézier curves. An ordered list $\mathbf{p}_0, \ldots, \mathbf{p}_d$ of affinely independent points in $\mathbb{R}^d$ determines an orientation of $\mathbb{R}^d$ – simply consider the basis $\mathbf{p}_1 - \mathbf{p}_0$, $\mathbf{p}_2 - \mathbf{p}_0$, . . ., $\mathbf{p}_d - \mathbf{p}_0$.

Let $\mathcal{A}$ and $\mathcal{B}$ be the finite sets of points in $\mathbb{R}^d$. Suppose that $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_d\}$ is an affinely independent subset of $\mathcal{A}$. If the corresponding subset $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d\}$ of $\mathcal{B}$ is also affinely independent, then each subset determines an orientation, and the two orientations are either the same or they are opposite. $\mathcal{A}$ and $\mathcal{B}$ are said to be **compatible** if either every pair of orientations is the same, or if every such pair of orientations is opposite.

In Figure 1, the first and second point sets are compatible, but neither is compatible with the third.

Actually, $\mathcal{A}$ and $\mathcal{B}$ may not lie in the same space, like curves. By the compatible definition, Craciun et al. proved the following conditions.

Lemma 1. *[31] Let $\mathcal{A} \subset \mathbb{R}$, $\omega = \{\omega_i > 0 | i = 0, 1, \ldots, m\}$, and $\mathcal{P} \subset \mathbb{R}^d (d \in \{2, 3\})$, be the lattice points, weights, and control points of $F_{\mathcal{A},\omega,\mathcal{P}}$ and Δ be the convex hull of $\mathcal{A}$. If there is a projection $\pi_d : \mathbb{R}^d \to \mathbb{R}$ such that $\mathcal{A}$ is compatible with the image $\pi_d(\mathcal{P})$ of $\mathcal{P}$, then the mapping $F^{\circ}_{\mathcal{A},\omega,\mathcal{P}} : \Delta \to \mathbb{R}^d$ is injective.*

The proof of Lemma 1 is in Theorem 3.7 on page 12 in reference [31] and its 2D supplementary proof in reference [32].

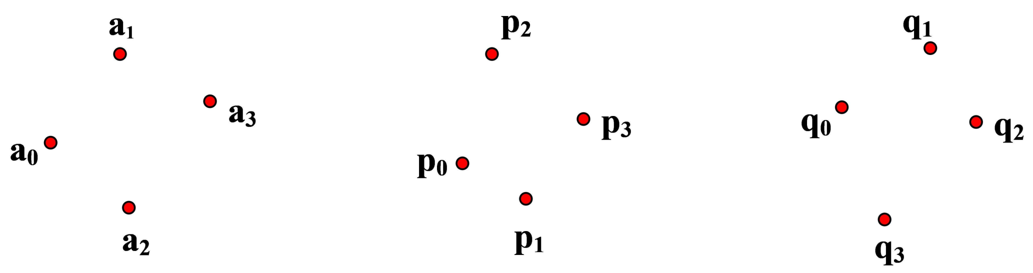


Figure 1. Compatible and incompatible points.

Remark 1. *$F^{\circ}_{\mathcal{A},\omega,\mathcal{P}}$ is the interior of curves.*

Definition 2. *If the boundary control points $\mathbf{p}_0$ and $\mathbf{p}_m$ do not coincide, and there exists a projection $\pi_d : \mathbb{R}^d \to \mathbb{R}$ such that $\mathcal{A}$ is compatible with the image $\pi_d(\mathcal{P})$ of $\mathcal{P}$, then the lattice points $\mathcal{A}$ and control points $\mathcal{P}$ are called strongly compatible.*

Given point sets $Q, Q', Q'' \subset \mathbb{R}^d$, it is easy to check the following relationships.

- Reflexivity. Q is compatible with itself.
- Transitivity. If Q and Q' are compatible and Q' and Q'' are compatible, then Q and Q'' are compatible.

Lemma 2. *[31] The map $F_{\mathcal{A},\omega,\mathcal{P}}$ is injective for all $\omega > 0$ if and only if $\mathcal{A}$ and $\mathcal{P}$ are strongly compatible.*

Proof. The lattice points $\mathcal{A}$ and control points $\mathcal{P}$ being strongly compatible is equivalent to $\mathcal{A}$ and $\mathcal{P}$ being compatible and $\mathbf{p}_0$, $\mathbf{p}_m$ do not coincide. By Lemma 1, $\mathcal{A}$ and $\mathcal{P}$ being compatible is equivalent to $F^{\circ}_{\mathcal{A},\omega,\mathcal{P}}$ being injective. As rational Bézier curves satisfy endpoint interpolation, $\mathbf{p}_0$, $\mathbf{p}_m$ not coinciding implies that the boundary of $F_{\mathcal{A},\omega,\mathcal{P}}$ is not self-intersecting. This completes the proof.

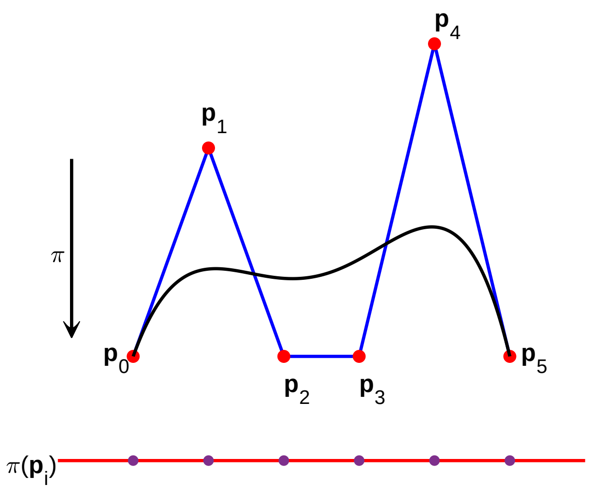


Figure 2. A compatible projection.

Example 1. *For the curve in Figure 2, the projection $\pi : \mathbb{R}^2 \to \mathbb{R}$ maps control points $\{\mathbf{p}_0, \dots, \mathbf{p}_5\}$ to the points on the line in the same order as the lattice points $\mathcal{A} = \{0, 1, 2, 3, 4, 5\}$, which implies the curve has no self-intersections. In this figure, the projection π is simply the vertical projection forgetting the second coordinate.*

To reduce the complexity of the algorithm for checking the injectivity of rational Bézier curves, we first introduce the concept of monotone chains. The concept of monotone chain a basic concept in computational geometry, which can be found in a lot of literatures, such as [37,39,40]. We borrow its definition from literature [37].

Definition 3. *[39] Given a direction, if a chain has only one intersection point with each line which is perpendicular to this direction, the chain would be called as monotone chain along this direction. The given direction is called scan-line direction, and the line perpendicular to the direction is called scan-line.*

In [37], the concept of monotone chain is explained in geometry. It implies that if the orthogonal projection on line of every coordinate point on the chain is orderly, the chain is monotony relative to the line [39]. The illustrations of monotone chains in this paragraph are borrow from [37]. Let P be a control polygon with vertices $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_m$ running counterclockwise around its boundary. It is obvious that $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_m$ are the control points. The sides of P, called arcs, are denoted as $\mathbf{e}_i = (\mathbf{p}_i, \mathbf{p}_{i+1})$ and are directed from $\mathbf{p}_i$ to $\mathbf{p}_{i+1}$. A chain $C = (\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_{m-1})$ is a sequence of arcs on the boundary of P. C is monotone with respect to a (straight) line l if the projections of the vertices $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_m$ on l are ordered the same as the vertices in C. Let θ_i be the counterclockwise polar angle at arc $\mathbf{e}_i$ ($i = 0, \dots, m-1$) with respect to a chosen direction. Define α_i as the counterclockwise wedge from θ_{i-1} to θ_i if the external angle at vertex $\mathbf{p}_i$ is greater than or equal to 180 degrees; as the clockwise wedge α_i, ($i = 1, \dots, m-1$) has size less than 180 degrees. Given a chain $C = (\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_{m-1})$, define $\alpha(C) \triangleq \bigcup_{i=1}^{m-1} \alpha_i$ is the union of the wedges $\alpha_1, \dots, \alpha_{m-1}$. Obviously, $\alpha(C)$ is a wedge. Then the following results hold.

Lemma 3. *[37] $C = (\mathbf{e}_1, \dots, \mathbf{e}_k)$ is monotone with respect to l if and only if the normal to l has a polar angle $\theta \notin \alpha(C)$.*

The proof of Lemma 3 can be found in Lemma 1 of reference [37]. We use Lemma 3 to check whether a chain is monotone in our algorithm. (See Algorithm 1).

Theorem 1. *Given lattice points $\mathcal{A} = \{0, 1, \dots, m\}$, control points $\mathcal{P} = \{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_m\} \subset \mathbb{R}^2$. The chain defined by $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_m$ in counterclockwise is denoted as $C = (\mathbf{e}_0, \dots, \mathbf{e}_{m-1})$. $\mathcal{A}$ and $\mathcal{P}$ are strongly compatible if and only if C is monotone.*

Proof. If C is monotone, then by the definition of monotone chain C, the projections of control points $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_m$ on l have the same or opposite order as $0, \dots, m$. The following results then hold.

- There are no control points coinciding with each other.
- There exists a projection $\pi : \mathbb{R}^2 \to \mathbb{R}$, such that image $\pi(\mathcal{P})$ have the same or opposite order as $\{0, 1, \dots, m\}$, which equal the lattice points $\mathcal{A}$.

This implies that $\mathbf{p}_0$ and $\mathbf{p}_m$ do not coincide and control points $\mathcal{P}$ are compatible with lattice points $\mathcal{A}$. Therefore, $\mathcal{P}$ is strongly compatible with lattice points $\mathcal{A}$.

Meanwhile, if $\mathcal{A} \subset \mathbb{R}$ and $\mathcal{P} \subset \mathbb{R}^2$ are strongly compatible, then by Definition 2, the following holds.

- $\mathbf{p}_0$ and $\mathbf{p}_m$ do not coincide.
- There is a projection $\pi : \mathbb{R}^2 \to \mathbb{R}$ such that $\mathcal{A}$ is compatible with the image $\pi(\mathcal{P})$ of $\mathcal{P}$.

$\pi(\mathbf{p}_i)$ $(i = 0, \ldots, m)$ then has the same or opposite order as $\{0, \ldots, m\}$. Meanwhile, the lattice points of Bézier curves lie on a line. This means that there exists a projection π such that image $\pi(\mathbf{p}_i)$ $(i = 0, \ldots, m)$ on a line have the same or the opposite order as $\{0, \ldots, m\}$. The chain $C = (\mathbf{e}_0, \ldots, \mathbf{e}_{m-1})$ is defined by $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_m$ in a counterclockwise direction. This means that projection π does not change the order of the vertices $\{\mathbf{p}_0, \ldots, \mathbf{p}_m\}$ when projecting them onto a line and that the chain C is monotone.

By the theory of order-preserving mappings, Lemma 4 is obvious.

Lemma 4. *Given lattice points $\mathcal{A} = \{0, 1, \ldots, m\}$, control points $\mathcal{P} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_m\} \subset \mathbb{R}^3$. If $\mathcal{A}$ and $\mathcal{P}$ are strongly compatible, there exists an order-preserving mapping $\phi : \mathbb{R}^3 \to \mathbb{R}^2$ such that $\mathcal{A}$ and image $\phi(\mathcal{P})$ are strongly compatible.*

Proof. By an order-preserving mapping $\phi : \mathbb{R}^3 \to \mathbb{R}^2$, it is obvious that $\phi(\mathbf{p}_i), i = 0, \ldots, m$ defines the same orders as $\mathbf{p}_i, i = 0, \ldots, m$. $\phi(\mathcal{P})$ is then compatible with $\mathcal{P}$. Therefore, $\phi(\mathcal{P})$ is compatible with $\mathcal{A}$. Moreover, $\mathbf{p}_0$ does not coincide with $\mathbf{p}_m$. Therefore, $\phi(\mathbf{p}_0)$ does not coincide with $\phi(\mathbf{p}_m)$. $\mathcal{A}$ and image $\phi(\mathcal{P})$ are then strongly compatible.

Theorem 2. *Given lattice points $\mathcal{A} = \{0, 1, \ldots, m\}$, control points $\mathcal{P} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_m\} \subset \mathbb{R}^3$ and an order-preserving mapping $\phi : \mathbb{R}^3 \to \mathbb{R}^2$. The chain defined by $\phi(\mathbf{p}_0), \phi(\mathbf{p}_1), \ldots, \phi(\mathbf{p}_m)$ in counterclockwise is denoted as $\phi(C) = (\phi(\mathbf{e}_0), \ldots, \phi(\mathbf{e}_{m-1}))$. $\mathcal{A}$ and $\mathcal{P}$ are strongly compatible if and only if $\phi(C)$ is monotone.*

Proof. If $\mathcal{A}$ and $\mathcal{P}$ are strongly compatible, then by Lemma 4, the image $\phi(\mathcal{P})$ under an order-preserving mapping $\phi : \mathbb{R}^3 \to \mathbb{R}^2$ is strongly compatible with $\mathcal{A}$. By Theorem 1, $\phi(\mathcal{P})$ and $\mathcal{A}$ are strongly compatible is equivalent to $\phi(C)$ is monotone. On the other hand, given an order-preserving mapping $\phi : \mathbb{R}^3 \to \mathbb{R}^2$ such that $\phi(C)$ is monotone. By Theorem 1, $\phi(\mathcal{P})$ is strongly compatible with $\mathcal{A}$. As ϕ is an order-preserving mapping, orientations $\phi(\mathbf{p}_0) - \phi(\mathbf{p}_1), \phi(\mathbf{p}_0) - \phi(\mathbf{p}_2), \ldots, \phi(\mathbf{p}_0) - \phi(\mathbf{p}_m)$ decided by $\phi(\mathbf{p}_i), i = 0, \ldots, m$ is the same as that decided by $\mathbf{p}_i, i = 0, \ldots, m$. By the definition of strongly compatible, $\mathcal{P}$ is strongly compatible with $\phi(\mathcal{P})$. Therefore, $\mathcal{P}$ and $\mathcal{A}$ are strongly compatible. This completes the proof.

By Theorem 1, we propose algorithms (see Algorithms 1–3) for checking the strong compatibility of the lattice points $\mathcal{A}$ and control points $\mathcal{P}$ of rational Bézier curves. Furthermore, by Lemma 2, the strong compatibility of the lattice points $\mathcal{A}$ and control points $\mathcal{P}$ of rational Bézier curves is equivalent to rational Bézier curves being injective.

Algorithm 1 is for computing wedge $\alpha(C)$. $\alpha(C) < 180^\circ$ is equivalent to chain C being monotone. It has been discussed in literature [37] that the complexity of this procedure is approximately $O(m)$.

Algorithm 1 Computing $\alpha(C)$

```
Input: control points $\mathcal{P} = \{\mathbf{p}_i | i = 0, \ldots, m\} \subset \mathbb{R}^2$
Output: $\alpha(C)$
1: Computing arcs of $P$, denoted as $C = (\mathbf{e}_1, \ldots, \mathbf{e}_{m-1})$
2: for all $\mathbf{e}_i \in C$ $(i = 0, \ldots, m-1)$ do
3:     computing polar angles $\theta_i$ of $\mathbf{e}_i$
4:     let $\theta = \{\theta_0, \ldots, \theta_{m-1}\}$.
5: end for
6: for all $\theta_i \in \theta$ $(i = 0, \ldots, m-2)$ do
7:     computing wedges $\alpha_i$ from $\theta_i$ to $\theta_{i+1}$
8:     let $\alpha = \{\alpha_0, \ldots, \alpha_{m-2}\}$.
9: end for
10: computing the $\alpha(C)$, union of the wedges $\alpha_0, \ldots, \alpha_{m-2}$
11: return $\alpha(C)$
```

By Algorithm 1, we propose an algorithm for checking the strong compatibility of lattice points $\mathcal{A}$ and control points $\mathcal{P}$, as shown in Algorithm 2.

Algorithm 2 Checking the strong compatibility of control points $\mathcal{P} \subset \mathbb{R}^2$ and lattice points $\mathcal{A}$

```
Input: control points $\mathcal{P} = \{\mathbf{p}_i | i = 0, \ldots, m\} \subset \mathbb{R}^2$ and lattice points $\mathcal{A} = \{0, 1, \ldots, m\}$.
Output: The compatibility of control points $\mathcal{P}$ and lattice points $\mathcal{A}$
1: if there exist two control points of $\mathcal{P}$ that are coincided then
2:     return $\mathcal{P}$ and $\mathcal{A}$ are not compatible
3: else   Computing wedges of control points $\alpha(C)$ by Algorithm 1
4:     if $\alpha(C) \geq 180^\circ$ then
5:         return $\mathcal{P}$ and $\mathcal{A}$ are not strongly compatible
6:     else
7:         return $\mathcal{P}$ and $\mathcal{A}$ are strongly compatible
8:     end if
9: end if
```

In Algorithm 2, Step 1 needs to check whether the two points coincide. The complexity of this procedure is approximately $O(m)$ (using Matlab). Step 3 requires $\alpha(C)$ to be computed. The complexity of this procedure is approximately $O(m)$ (as mentioned in Algorithm 1). Therefore, the complexity of Algorithm 2 is approximately $O(m)$, which deduces the complexity $O(m^4)$ of the algorithm in reference [33] a lot.

In Algorithm 3, Step 1 needs to check whether the two points coincide. The complexity of this procedure is approximately $O(m)$ (using Matlab). Steps 4 to 9 require the algorithm to go through all the points once in computing normal vectors and projections. This process has complexity of approximately $O(m)$. Step 10 requires $\alpha(C)$ to be computed. The complexity of this procedure is approximately $O(m)$. Therefore, the complexity of Algorithm 3 is approximately $O(m)$.

In Table 1, we present the computing time of Algorithms 2 and 3 for injective and non-injective

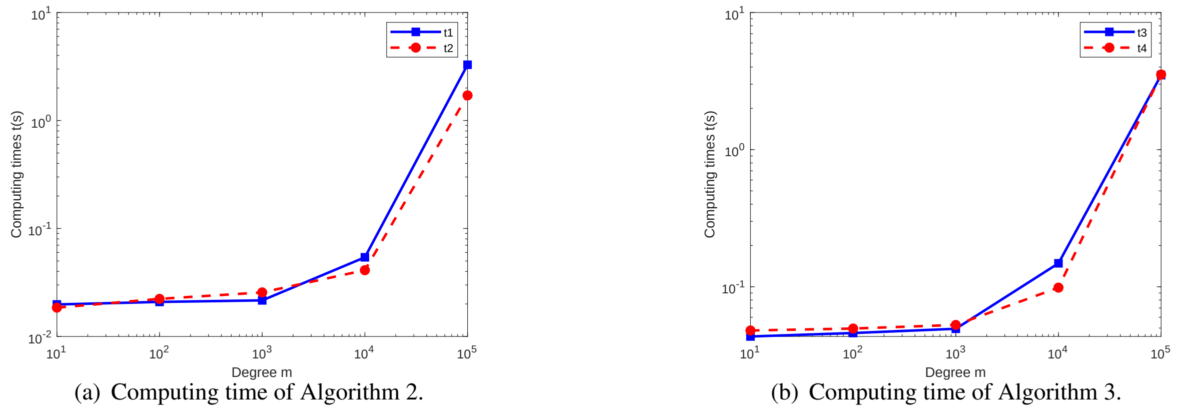
Table 1. Computing time of algorithms.

	m	10	10^2	10^3	10^4	10^5
Algorithm 2	t_1(s)	0.0198	0.0209	0.0216	0.0541	3.2816
	t_2(s)	0.0185	0.0223	0.0256	0.0411	1.7086
Algorithm 3	t_3(s)	0.0434	0.0460	0.0494	0.1483	3.4961
	t_4(s)	0.0480	0.0496	0.0527	0.0986	3.5299
Algorithm in [33]	t_5(s)	0.0976	10971	–	–	–
	t_6(s)	0.0290	0.0309	0.5712	56.2309	5970.8

[1] m is the number of control points of Bézier curves, t_1, t_3 and t_5 are the times required for computing injective curves, and t_2, t_4 and t_6 are the times required for computing non-injective curves.

[2] "–" indicates that the computing time exceeds 3 hours.

rational Bézier curves of degree $m = 10, 10^2, 10^3, 10^4$ and 10^5. Meanwhile, we make a comparison with an algorithm from the literature [33]. In Table 1, m is the number of control points of Bézier curves, t_1, t_3 and t_5 are the times required for computing injective curves, and t_2, t_4 and t_6 are the times required for computing non-injective curves. Our algorithms clearly take less time to run. When m exceeds 10^2, our algorithms run quickly, whereas the algorithm from [33] takes more than 3 hours to run. Additionally, we also present computing times of Algorithms 2 and 3 in Figure 3. Each result given in Table 1 is the average for 10 experiments conducted. All experiments are implemented by Matlab 2020b running on 3.20GHz, Intel(R) Core(TM) i7-8700U CPU with 8 GB RAM.



(a) Computing time of Algorithm 2.

(b) Computing time of Algorithm 3.

Figure 3. Computing time of Algorithms.

Algorithm 3 Checking the strong compatibility of control points $\mathcal{P} \subset \mathbb{R}^3$ and lattice points $\mathcal{A}$

```
Input: control points $\mathcal{P} = \{\mathbf{p}_i | i = 0, \ldots, m\} \subset \mathbb{R}^3$ and lattice points $\mathcal{A} = \{0, 1, \ldots, m\}$.
Output: The compatibility of control points $\mathcal{P}$ and lattice points $\mathcal{A}$
1: if there exist two (or several) control points of $\mathcal{P}$ that coincide then
2:     return $\mathcal{P}$ and $\mathcal{A}$ are not compatible
3: else
4:     Select an order-preserving projection $\phi$
5:     for $i = 0 \to m$ do
6:     Calculate the projection $\phi(\mathbf{p}_i)$ of $\mathbf{p}_i$
7:     Calculate the plane $S$ on which $\phi(\mathbf{p}_i)$ $i = 0, \ldots, m$ lie.
8:     end for
9:     Rotate $S$ to $S'$, which is parallel to the plane $z = 0$
10:    Computing wedges $\alpha(C)$ of the points $\phi(\mathbf{p}_i)$ by Algorithm 1
11:    if $\alpha(C) \geq 180°$ then
12:        return $\mathcal{P}$ and $\mathcal{A}$ are not strongly compatible
13:    else
14:        return $\mathcal{P}$ and $\mathcal{A}$ are strongly compatible
15:    end if
16: end if
```

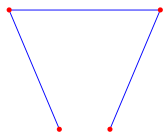
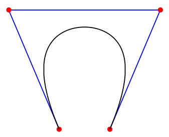
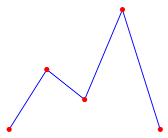
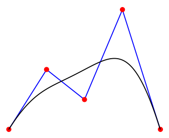
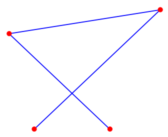
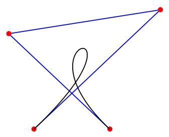
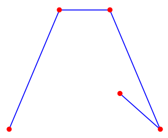
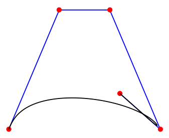
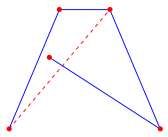
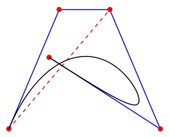
3. Examples

In this section, we illustrate some numerical examples of checking the injectivity of rational Bézier curves by Algorithms 2 and 3. Some two-dimensional examples are considered in Example 2. Some three-dimensional examples are considered in Example 3. An interesting example, a rabbit composed by five rational Bézier curves, is shown in Example 4. All the experiments in this section are conducted using an Intel(R) Core(TM) i7-8700U CPU @ 3.20GHz 3.19GHz.

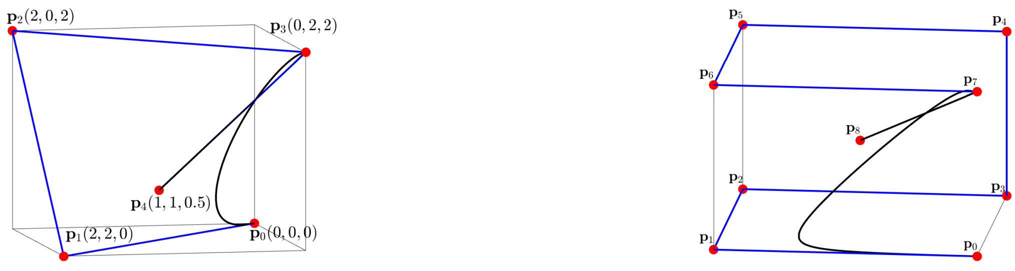
Example 2. *We consider five 2D rational Bézier curves. The control points, control polygons, and curves are given in Table 2. By Definition 2, we can check whether the control points and lattice points are strongly compatible. By Theorem 1, we can judge their injectivity. The results are given in Table 2. The first column presents each control point and each control polygon. The second column presents the degree m. The third column presents the strong compatibility. The forth column presents the computing time when using Algorithm 2 to check whether the control points are strongly compatible with the lattice points. The computing time is measured in seconds. The fifth column presents rational Bézier curves together with weights. The sixth column presents the injectivity of the corresponding rational Bézier curves. The numerical experiments show that the use of Algorithm 2 is effective and feasible.*

Example 3. *Consider two 3D rational Bézier curves of degrees* 4 *and* 8. *Using Algorithm 3, it takes* $0.01s$ *and* $0.012s$ *to determine that the control points are both not strongly compatible with their lattice points. Therefore, these two rational Bézier curves are not injective by Theorem 1. We can find these two curves be self-intersecting with weights* $1, 1, 1, 100, 1$ *and* $10, 10, 1, 1, 1, 1, 1, 100, 1$. *The control points and the self-intersecting rational Bézier curves are shown in Figure 4.*

Table 2. Two-dimensional rational Bézier curves.

Control points	Degree	strongly compatible	Time(s)	Curves	Injectivity
	3	Yes	0.021	 $\omega = \{1, 2, 2, 1\}$	Yes
	4	Yes	0.018	 $\omega = \{1, 2, 2, 2, 1\}$	Yes
	3	No	0.02	 $\omega = \{1, 1, 1, 1\}$	No
	4	No	0.021	 $\omega = \{2, 1, 1, 30, 2\}$	No
	4	No	0.021	 $\omega = \{1, 1, 2, 5, 1\}$	No

Example 4. *In this example, we use five rational Bézier curves drawing a "rabbit-shape" as shown in Figure 5. The control points, control polygons, weights, and the rational Bézier curves are shown in Figure 5. We check the strong compatibility of each control points and lattice points by Algorithm 2. The computing times are* 0.039*,* 0.0056*,* 0.0372*,* 0.0033*,* 0.0367*s. The coordinates of five rational Bézier curves in "rabbit-shape" is shown in Table 3.*

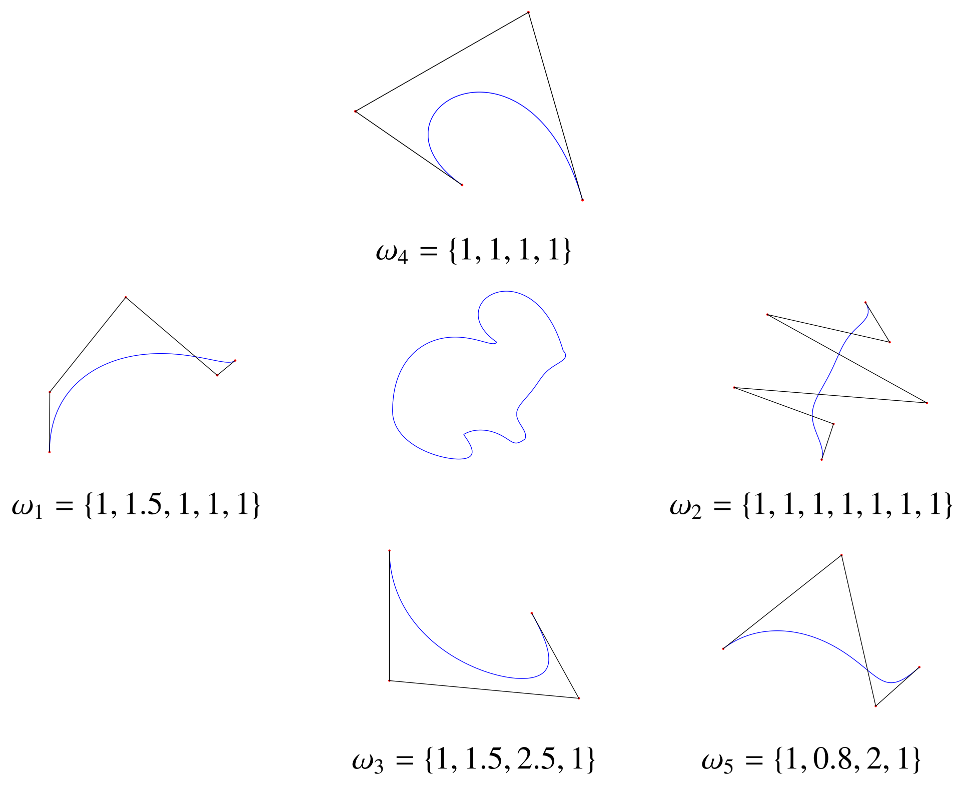


(a) Self-intersecting 3D rational Bézier curve of degree 4. (b) Self-intersecting 3D rational Bézier curve of degree 8.

Figure 4. Two self-intersecting 3D rational Bézier curves.

Table 3. Coordinates of control points of five rational Bézier curves in "rabbit-shape".

Bézier curves	Coordinates of control points			
$c_1(t)$	(−0.6230, −0.3300)	(−0.6217, 0.0820)	(−0.3138, 0.7315)	(0.0570, 0.1966)
	(0.1298, 0.2982)			
$c_2(t)$	(0.3312, −0.5862)	(0.4061, −0.4024)	(−0.2202, −0.2154)	(0.9951, −0.2946)
	(−0.0112, 0.1600)	(0.7604, 0.0177)	(0.60744, 0.22224)	
$c_3(t)$	(−0.6230, −0.3300)	(−0.6234, −0.7756)	(0.0595, −0.8362)	(−0.1105, −0.5442)
$c_4(t)$	(0.1298, 0.2982)	(−0.2931, 0.6686)	(0.3928, 1.1666)	(0.6074, 0.2222)
$c_5(t)$	(−0.1105, −0.5442)	(0.1559, −0.3302)	(0.2329, −0.6752)	(0.3312, −0.5862)



$\omega_4 = \{1, 1, 1, 1\}$
$\omega_1 = \{1, 1.5, 1, 1, 1\}$
$\omega_2 = \{1, 1, 1, 1, 1, 1, 1\}$
$\omega_3 = \{1, 1.5, 2.5, 1\}$
$\omega_5 = \{1, 0.8, 2, 1\}$

Figure 5. A "rabbit" composed by five rational Bézier curves.

4. Conclusions

In this paper, we have presented algorithms for checking the injectivity of rational Bézier curves in $\mathbb{R}^d, d \in \{2, 3\}$. Some typical examples have been used to show that the proposed algorithms are effective. The complexity and computing time of these two algorithms have also been derived. In summary, two effective and time-saving algorithms with a complexity of just $O(m)$ have been proposed, representing a significant improvement on the previous algorithm [33], which has $O(m^5)$ complexity for spacial case and $O(m^4)$ for 2D cases. Zhao and Zhu [41, 42] studied the sufficient and necessary conditions of injectivity of NURBS curves and surfaces. Yu et al. [35] studied the geometric conditions of injectivity of toric volumes. Zhao et al. [36] studied sufficient and necessary conditions of injectivity of Bézier volumes. To generalize our algorithm to check the injectivity of NURBS curves and surfaces, and volumes is an interesting work for us in future. Moreover, applying our algorithms in computing non-self-intersecting offset curves and surfaces is another interesting work for us in future.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant numbers 11801053, 12071057) and the Fundamental Research Funds for the Central Universities (grant numbers 3132022203).

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. W. Boehm, H. Prautzsch, *Geometric Foundations of Geometric Design*, AK Peter, Boston, 1992.
2. G. Farin, *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, Academic Press, 1993.
3. G. E. Farin, D. Hansford, *The Essentials of CAGD*, A K Peters/CRC Press, 2000. http://doi.org/10.2307/3621635
4. J. Hoschek, D. Lasser, *Grundlagen der geometrischen Datenverarbeitung*, B. G. Teubner Stuttgart, 1992.
5. M. Duncan, *Applied Geometry for Computer Graphics and CAD*, Springer-Verlag, 2005.
6. M. Mortenson, *Geometric modeling*, New York: Wiley Computer Publishing, 2006.
7. L. A. Piegl, W. Tiller, *The NURBS book*, Springer-Verlag, 1997.
8. A. Rockwood, P. Chambers, Interactive curves and surfaces, in *Technology-Based Re-Engineering Engineering Education Proceedings of Frontiers in Education FIE'96 26th Annual Conference*, 1996, 471–474.
9. F. P. Preparata, M. I. Shamos, *Computational Geometry*, Springer-Verlag, 1985.
10. P. E. Bézier, Example of an existing system in the motor industry: The Unisurf system, *Proc. Roy. Soc. A*, **321** (1971), 207–218. http://doi.org/10.1098/rspa.1971.0027

11. P. Casteljau, De Casteljau's autobiography: My time at Citroën, *Comput. Aided Geom. Design*, **16** (1999), 583–586. http://doi.org/10.1016/S0167-8396(99)00024-2
12. G. Farin, Some aspects of car body design at Daimler-Benz, in *North-Holland Publ Co*, (1983), 93–97.
13. H. J. Hochfeld, M. Ahlers, Role of Bézier curves and surfaces in the Volkswagen CAD approach from 1967 to today, *Comput. Aided Design*, **22** (1990), 598–608. http://doi.org/10.1016/0010-4485(90)90045-E
14. P. Bézier, *Numerical control: mathematics and applications*, Jhon Wiley& Sons Ltd., Bristol, 1972.
15. P. Bézier, *The Mathematical Basis of the UNISURF CAD System*, Butterworth-Heinemann, 1986.
16. A. R. Forrest, Interactive interpolation and approximation by Bézier polynomials, *Comput. Aided Design*, **22** (1990), 527–537. http://doi.org/10.1016/0010-4485(90)90038-e
17. W. J. Gordon, R. F. Riesenfeld, Bernstein-Bézier methods for the computer-aided design of free-form curves and surfaces, *J. ACM*, **21** (1974), 293–310. http://doi.org/10.1145/321812.321824
18. G. Farin, Algorithms for rational Bézier curves, *Comput. Aided Design*, **15** (1983), 73–77.
19. G. Farin, J. Hoschek, M. S. Kim, *Handbook of Computer Aided Geometric Design*, Elsevier, 2002.
20. P. J. Barry, R. N. Goldman, Three examples of dual properties of Bézier curves, *Math. Methods Comput. Aided Geom. Design*, (1989), 61–69.
21. W. Boehm, On cubics: A survey, *Comput. Graph. Image Process.*, **19** (1982), 201–226. http://doi.org/10.1016/0146-664X(82)90009-0
22. D. Nairn, J. Peters, D. Lutterkort, Sharp, quantitative bounds on the distance between a Bézier curve and its control polygon, *Comput. Aided Geom. Design*, **16** (1999), 613–631. http://doi.org/10.1016/S0167-8396(99)00026-6
23. H. Lin, *Geometric iterative methods and their applications*, Zhejiang University, 2021.
24. H. Lin, T. Maekawa, C. Deng, Survey on geometric iterative methods and their applications, *Comput. Aided Design*, **95** (2018), 40–51. http://doi.org/10.1016/j.cad.2017.10.002
25. J. Hoschek, Offset curves in the plane, *Comput. Aided Design*, **17** (1985), 77–82. http://doi.org/10.1016/0010-4485(85)90249-0
26. Y. S. Chua, Bézier brushstrokes, *Comput. Aided Design*, **22** (1990), 550–555. http://doi.org/10.1016/0010-4485(90)90040-J
27. H. N. Phien, N. D. Ejdumrong, Efficient algorithms for Bézier curves, *Comput. Aided Geom. Design*, **17** (2000), 247–250. http://doi.org/10.1016/S0167-8396(99)00048-5
28. C. M. Hoffmann, *Geometric and solid modeling : An introduction*, Morgan Kaufmann, 1989.
29. D. Lasser, Calculating the self-intersections of Bézier curves, *Comput. Ind.*, **12** (1988), 259–268.
30. W. Tiller, E. G. Hanson, Offsets of two-dimensional profiles, *IEEE Comput. Graph. Appl.*, **4** (1984), 36–46. http://doi.org/10.1109/MCG.1984.275995
31. G. Craciun, L. Garcia-Puente, F. Sottile, Some geometrical aspects of control points for toric patches, *Lect. Notes Comput. Sci.*, **5862** (2010), 111–135.

32. F. Sottile, C. G. Zhu, Injectivity of 2D toric Bézier patches, in *International Conference on Computer-aided*, (2011), 235–238.

33. C. Zhu, X. Zhao, Self-intersections of rational Bézier curves, *Graph. Models*, **76** (2014), 312–320. http://doi.org/10.1016/j.gmod.2014.04.001

34. Y. Yu, Y. Ji, C. Zhu, An improved algorithm for checking the injectivity of 2D toric surface patches, *Comput. Math. Appl.*, **79** (2020), 2973–2986. http://doi.org/10.1016/j.camwa.2020.01.001

35. Y. Yu, Y. Ji, C. Zhu, Conditions for injectivity of toric volumes with arbitrary positive weights, *Comput. Graph.*, **97** (2021), 88–98. http://doi.org/10.1016/j.cag.2021.04.026

36. X. Zhao, J. Li, S. He, C. Zhu, Geometric conditions for injectivity of 3D Bézier volumes, *AIMS Math.*, **6** (2021), 11974–11988. http://doi.org/10.3934/math.2021694

37. F. P. Preparata, K. J. Supowit, Testing a simple polygon for monotonicity, *Inf. Process. Lett.*, **12** (1981), 161–164. http://doi.org/10.1016/0020-0190(81)90091-0

38. R. Krasauskas, Toric surface patches, *Adv. Comput. Math.*, **17** (2002), 89–113. http://doi.org/10.1023/A:1015289823859

39. Y. Zhou, A new algorithm for polygon intersection operation, in *International Conference on Computer & Electrical Engineering*, 2009. http://doi.org/10.1109/ICCEE.2009.19

40. S. Park, H. Shin, Polygonal chain intersection, *Comput. Graph-UK*, **26** (2002), 341–350. http://doi.org/10.1016/S0097-8493(02)00060-2

41. X. Zhao, C. Zhu, Injectivity of NURBS curves, *J. Comput. Appl. Math.*, **302** (2016), 129–138. http://doi.org/10.1016/j.cam.2016.01.046

42. X. Zhao, C. Zhu, H. Wang, Geometric conditions of non-self-intersecting NURBS surfaces, *Appl. Math. Comput.*, **310** (2017), 89–96. http://doi.org/10.1016/j.amc.2017.04.016