Visual Comput (2006)
DOI 10.1007/s00371-006-0075-6

ORIGINAL ARTICLE

Ming Li
Xiao-Shan Gao
Shang-Ching Chou

Quadratic approximation to plane parametric curves and its application in approximate implicitization*

X.-S. Gao (✉)
Key Lab of Mathematics Mechanization, Academia Sinica, China
xgao@mmrc.iss.ac.cn

M. Li
School of Computer Science, Cardiff University, UK

S.-C. Chou
Department of Computer Science, Wichita State University, USA

Abstract Expressing complex curves with simple parametric curve segments is widely used in computer graphics, CAD and so on. This paper applies rational quadratic B-spline curves to give a global C^1 continuous approximation to a large class of plane parametric curves including rational parametric curves. Its application in approximate implicitization is also explored. The approximated parametric curve is first divided into intrinsic *triangle convex segments* which can be efficiently approximated with rational quadratic Bézier curves. With this approximation, we keep the convexity and the cusp (sharp) points of the approximated curve with simple computations. High accuracy approximation is achieved with a small number of quadratic segments. Experimental results are given to demonstrate the operation and efficiency of the algorithm.

Keywords Rational approximation · Conics · Parametric curves · Approximate implicitization

1 Introduction

Conics or rational quadratic curves have many simple and classical properties, and they are widely studied and applied in various fields such as computer graphics, CAD, image processing and so on [11, 13, 16, 21, 23, 24]. Approximately expressing a plane parametric curve possibly in transcendental form with conics can import various complex curves into these fields and facilitate their applications. In this paper, we propose an efficient method to approximate plane parametric curves with rational quadratic splines. Its application in approximate implicitization is also explored.

Many methods have been proposed to approximate plane curves using parametric curves in low degree, including high accuracy approximation [5, 9], points sampling approximation [22, 30, 31], various approximations using polynomials [29], rational curves [2] and linear segments [7]. The high accuracy approximation was introduced by de Boor [5] and then improved in later work [9]. This work is mainly concerned with the local approximation effect at a point on a given curve. For rational functions, Wang et al. investigated the necessary and sufficient convergence criteria for their polynomial approximations, based on the notion of hybrid polynomials [29]. Bajaj and Xu, on the other hand, applied a local expansion method to approximate an implicit algebraic curve with C^1 continuous piecewise rational curves [2]. Closely related to this work, a systematic method for linear approximation

*Partially supported by a Chinese NKBRPC grant 2004CB318000, an EPSRC grant GR/S69085/01, and a USA NSF grant CCR-0201253.

of composite plane and space Bézier curves is due to Cho et al. [7]. In the work, significant points are first extracted from an approximated curve based on its curvature and torsion variation, providing an initial rough approximation, which is then further subdivided for error control and topology consistency between the curve and its approximation.

Approximating plane curves using conics has also been widely studied [1, 11, 20, 21, 23, 31]. Under certain error expression, the necessary and sufficient condition to optimally approximate a plane curve using conics was given in [1]. Curvature continuous approximation methods to plane curves have been proposed by Farin [11], Pottmann [21], and recently by Yang [31]. Most of the approximation methods, however, mainly consider certain curve segments possessing some simple properties, e.g., without inflection points (flexes for short) inside or lying within a triangle. How to get such segments from various general complex curves, e.g., curves in Figs. 1 or 2, is not fully addressed. For the specific application of conics in font plotting, an algorithm is provided to approximate a general parametric curve using the split-and-merger method for adaptively choosing knots of the approximation splines [13], where iterative sampling of points and merging of approximation curve segments are applied so that finally each of these resulted segments lies in a triangle. However, no specific error control method is provided in this work.

Another motivation of our work is approximate implicitization. Given a rational parametric curve, we can always convert it into implicit form, which is called implicitization. But for a general parametric curve $\mathcal{C}(t) = (x(t), y(t))$ where $x(t)$ and $y(t)$ may be non-rational functions such as trigonometric and exponential functions, we usually cannot compute its exact implicit form. Even if the exact implicit form can be computed, it is not necessary to do so in many cases. The reason is that some complicated computations may be involved in the process of exact implicitization and the resulted implicit form can have large numbers as coefficients, e.g., the curves in Sect. 5. Furthermore, as investigated by Sederberg, an exact implicitization form may have self-intersections or some unwanted branches [26]. All these unexpected properties limit the applications of the exact implicitization in practical fields.

Due to these reasons, the idea of approximate implicitization has been proposed. Power series sequences are applied by Montaudouin et al. to give local explicit approximations for curves and surfaces around a singular point [18]. The method is extended by Chuang and Hoffmann to give a local implicit approximation to a parametric curve or surface [8]. A systematic work to implicitly approximate parametric curves or surfaces is proposed by Dokken using singular value decomposition to find a set of alternative approximations [10]. Sederberg et al. considered monoid curves and surfaces to find an approximate implicit equation and an approximate inversion map of a plane rational parametric curve or a rational parametric surface [26]. Shalaby et al. generated a C^1 continuous quadratic B-spline approximation for a parametric curve via its orthogonal projection in Sobolev spaces, and wavelets are then applied to reduce the curve segments [27]. Since the implicit form of conics can be easily obtained, if we can approximate a parametric curve with rational quadratic splines, its approximate implicitization is then a natural consequence. Furthermore, our algorithm ensures that each resulted approximation conics lies in a convex triangle, which confine the range of its implicit representation, and therefore intersections between unwanted branches outside the triangles will not be produced.

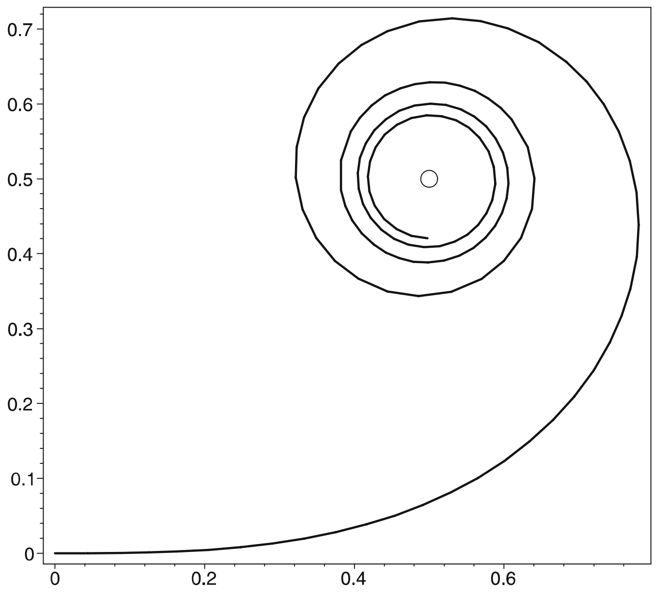


Fig. 1. Euler's spiral $\mathcal{C}_0$

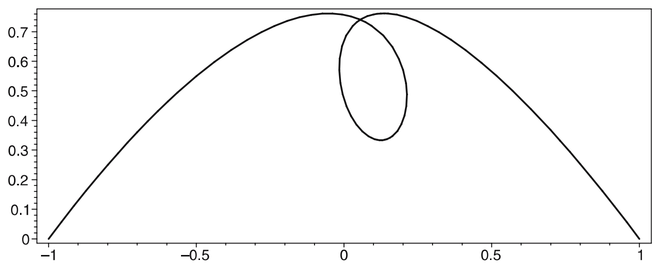


Fig. 2. An algebraic curve $\mathcal{C}_1$

Although these previous approximation methods for curve approximation perform very well in many cases, there are still important issues that are not addressed in detail. The first issue is the curve segmentation. Dividing a curve on its singular points is mentioned in some previous work. However, consider Euler's spiral $\mathcal{C}_0(t)$ [25] and an algebraic curve $\mathcal{C}_1(t)$ [1]:

$$\mathcal{C}_0(t) = \left(\int_0^t \cos\left(\frac{\pi}{2}\xi^2\right) d\xi, \int_0^t \sin\left(\frac{\pi}{2}\xi^2\right) d\xi \right),\ 0 \le t \le 4;$$

$$\begin{aligned}\mathcal{C}_1(t) = {} & (1 - 8t + 26t^2 - 32t^3 + 13t^4)b_0 - 4(-2t + 9t^2 \\ & - 14t^3 + 7t^4)b_1 + (10t^2 - 24t^3 + 15t^4)b_2,\end{aligned}$$

where $b_0 = (-1, 0),\ b_1 = (1, 4/3),\ b_2 = (1, 0)$, and $0 \le t \le 1$. The corresponding graphs of $\mathcal{C}_0(t)$ and $\mathcal{C}_1(t)$ are plotted in Figs. 1 and 2. Neither of these two curve segments have singular points. Therefore no segmentations are needed if considering singular points only. But most previous methods do not give good approximations for them even if a high order of approximation can be achieved. Actually it is implicitly assumed in these methods that the approximated curve segments are convex, but no detail is illustrated on how to get such convex segments from a general complex curve. In Fig. 1, for example, even if the curve is separated at its singular points and flexes (it does not have actually), the convexity of the resulted segments cannot yet be ensured. The second issue is how to keep the intrinsic geometric properties of the curve such as cusp points, convexity, etc. In this paper, these issues are to be resolved with an intrinsic segmentation of the approximated curve.

In this paper, we propose a geometric method to approximate a general complex plane parametric curve segment. The method consists of three steps. First, the curve to be approximated is divided into *triangle convex segments* by separating them at the cusp points, the flexes, the parallel points and shoulder points (definitions in Sect. 3). This curve segmentation does not depend on the coordinate and is intrinsic. The triangle convexity of the resulted segment makes it possible to have an efficient conics approximation using a global approximation method called *shoulder point approximation*. The approximate quadratic spline is finally converted into a rational quadratic B-spline with proper knot selection [4, 19]. The final quadratic B-spline obtained with our method keeps the convexity and the cusp (sharp) points of the approximated parametric curve. Experimental results show that high accuracy of approximation may be achieved with a small number of quadratic segments. In our method, the main computation is to solve univariate equations, for which there exist mature algorithms, ensuring generation of the final approximate spline in an efficient way. Furthermore, the method can be used to find approximations not only for rational parametric curves but also for a large class of parametric curves defined by non-rational functions such as trigonometric and exponential functions.

The rest of the paper is organized as follows. We introduce the rational quadratic Bézier curves as well as some of their properties in Sect. 2. Our main result, the approximation method, is illustrated in Sects. 3, 4 and 5. We conclude the paper in Sect. 6.

2 Rational quadratic Bézier curves

In the section, rational quadratic Bézier curves are introduced as well as some of their properties to be used in the paper.

Any rational quadratic segment or conics can be expressed by a *rational quadratic Bézier curve* in the following form [11, 16, 21]:

$$P(t) = \frac{P_0\phi_0(t) + \omega P_1\phi_1(t) + P_2\phi_2(t)}{\phi_0(t) + \omega\phi_1(t) + \phi_2(t)},\ 0 \le t \le 1, \tag{1}$$

where $\phi_0 = (1-t)^2,\ \phi_1 = 2t(1-t),\ \phi_2 = t^2$ are quadratic Bernstein basis; $\omega \in \mathcal{R}$ is *weight*, $P_i = (x_i, y_i) \in \mathcal{R}^2$ are *control points* of $P(t)$ and triangle $P_0P_1P_2$ is its *control triangle*. A rational quadratic Bézier curve in form (1) has the following properties [11, 16]:

(P1) *Convex hull:* segment $P(t),\ 0 \le t \le 1$, lies in the convex hull, i.e., the control triangle $P_0P_1P_2$, for $\omega > 0$.

(P2) *Endpoints interpolation:* from

$$\begin{aligned} P(0) &= P_0, & P(1) &= P_2, \\ P'(0) &= 2\omega(P_1 - P_0), & P'(1) &= 2\omega(P_2 - P_1), \end{aligned}$$

it can be seen that $P(t)$ passes through the endpoints P_0, P_2 and the two lines passing through points P_0, P_2 with direction $P'(0), P'(1)$ meet at point P_1.

(P3) *Type parameter:* the quadratic type of $P(t)$ is uniquely determined by its weight ω: for $0 < \omega < 1$, we have an ellipse segment; for $\omega = 1$, a parabola; and for $\omega > 1$, a hyperbola. If the sign of ω is changed, we get a complementary segment(s) of the conics, which is the other part(s) of the quadratic curve outside the control triangle (see Fig. 3).

(P4) *Shoulder point:* point $S = P(\frac{1}{2})$ is called the shoulder point of $P(t)$, which can be computed as

$$S = \frac{1}{2}(Q_0 + Q_2), \tag{2}$$

where $Q_0 = \frac{P_0 + \omega P_1}{1+\omega},\quad Q_2 = \frac{\omega P_1 + P_2}{1+\omega}$. The tangent line of $P(t)$ at the shoulder point S passes through Q_0 and Q_2 and is parallel to line P_0P_2. Furthermore, S is the unique point on $P(t),\ 0 \le t \le 1$, that has maximum distance to line P_0P_2, as illustrated in Fig. 4.

(P5) *Implicit form:* the implicit form of $P(t) = (x(t), y(t))$ in Eq. 1 is given by

$$\tau_1^2 = 4\omega^2\tau_0\tau_2, \tag{3}$$

where τ_i are barycentric coordinates of a point with respect to the control triangle $P_0P_1P_2$ with $P_i = (x_i, y_i),\ i =$

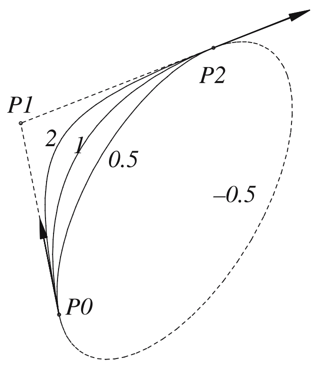


Fig. 3. Type parameters

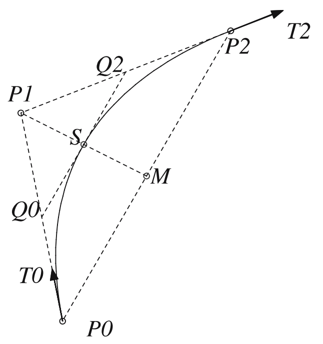


Fig. 4. Shoulder point

0, 1, 2. We have the following relation of the barycentric coordinate (τ_0, τ_1, τ_2) of a point in corresponding Cartesian coordinates (x, y)

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \tau_0 \\ \tau_1 \\ \tau_2 \end{pmatrix}$$

Thus we can get the implicit form of conics $P(t)$ from computing $\tau_i,\ i = 0, 1, 2$ from (x, y) using Cramer's rule.

3 Dividing a curve segment into triangle convex segments

In this section, assumptions of input curve segments to be approximated are first made. Then some basic concepts and definitions of a parametric curve are introduced. After these preparations, we go deep into the topic as how to divide a parametric curve segment into triangle convex segments, which can then be efficiently approximated by conics using the method described in Sect. 4.

The following notations and definitions are used in this paper.

For two vectors $V_i = (u_i, v_i),\ u_i, v_i \in \mathcal{R},\ i = 0, 1$, we have their dot product $V_0 \cdot V_1 = u_0 u_1 + v_0 v_1$ and cross product $V_0 \times V_1 = u_0 v_1 - u_1 v_0$.

The input curve segment $\mathcal{C}(t) = (x(t), y(t)),\ a \le t \le b$ is always assumed to be a curve segment such that for any $a \le t \le b,\ x'(t), x''(t), y'(t), y''(t)$ always exist. Furthermore, it is also assumed that for $a \le t \le b$, the slope $\lim_{s \to t} y'(s)/x'(s)$ either exists or approaches to infinity. For a curve segment $\mathcal{C}(t) = (x(t), y(t)),\ a \le t \le b$, the following definitions and symbols are given.

We call $P_0 = \mathcal{C}(a)$ the *left endpoint*, $T_- = \lim_{t \to a^+} (x'(t), y'(t))$ the corresponding *left tangent direction*, and the line passing through P_0 with direction T_- is the *left tangent line*. The *right endpoint* $P_2 = \mathcal{C}(b)$, the *right tangent direction* T_+ and the *right tangent line* can be defined in a similar way. With these definitions, $\mathcal{C}(t),\ a \le t \le b$ is sometimes denoted as $S[P_0, P_2]$ to show its left endpoint P_0 and right endpoint P_2 or $S[P_0, T_-, P_2, T_+]$ when its left and right tangent directions T_- and T_+ are also prescribed.

A point $\mathcal{C}(t_0)$ is called a *cusp point* or *singular point* if $x'(t_0) = y'(t_0) = 0$. A cusp point is usually a sharp point on the curve as shown in Fig. 5. The tangent direction at a cusp point $C(t_0)$ is defined as follows. Let $s = \lim_{t \to t_0} y'(t)/x'(t)$. If s is a finite number, we define $T = (1, s)$. If s approaches to infinity, we define $T = (0, 1)$. An *inflection point* $\mathcal{C}(t_0)$, flex for short, is a non-cusp point at which the sign of the curvature changes. Flexes can be found by solving the equation $x'(t)y''(t) - x''(t)y'(t) = 0$.

A point $\mathcal{C}(t_0)$ is called a *parallel point* if there exists $s_0 \ge a$ such that (1) point $\mathcal{C}(s_0)$ is a cusp point, a flex, or a boundary point; (2) there exist no cusp or flexes on $\mathcal{C}(t)$ for $s_0 < t < t_0$; and (3) the tangent directions at $C(s_0)$ and $C(t_0)$ are parallel. Suppose that the left tangent directions of $S[\mathcal{C}(s_0), \mathcal{C}(t_0)]$ is T_-. Then t_0 can be found by solving

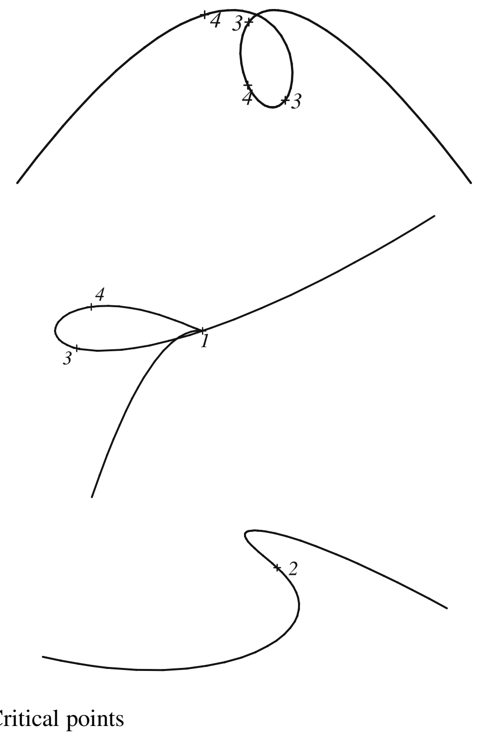


Fig. 5. Critical points

the following univariate equations:

$$T_- \times \mathcal{C}'(t) = 0,\ s_0 < t < s_1. \tag{4}$$

A curve segment $S[P_0,\, T_-,\, P_2,\, T_+]$ is said to be *triangle convex* if the left tangent line and the right tangent line meet at a point P_1 and the line segment P_0P_2 and S form a convex region inside the *control triangle* $P_0P_1P_2$ of S.

A point S_P on $S[P_0,\, T_0,\, P_2,\, T_2]$ is said to be a *shoulder point* if S_P has the maximal distance to the line P_0P_2.

Lemma 1. *If a curve segment $S[P_0,\, P_2]$ is triangle convex and does not contain a segment of straight line, then $S[P_0,\, P_2]$ has a unique shoulder point.*

Proof. Suppose that there are two shoulder points S_1 and S_2. We can see that the line segment S_1S_2 should be inside the region formed by line P_0P_2 and S. Since S_1 and S_2 have maximal distance to P_0P_2, the line segment S_1S_2 must be coincident with S, a contradiction. □

From Lemma 1, the shoulder point for a triangle convex segment $\mathcal{C}(t), a \le t \le b$, can be computed using Newton–Ralphson method by solving the following equation with an initial value $t = (a+b)/2$,

$$(x'(t),\, y'(t)) \times (P_2 - P_0) = 0. \tag{5}$$

The curve segmentation in this paper will finally divide an input approximated parametric curve into several triangle convex segments, separated by four types of *intrinsic critical points*: the cusp points, the flexes, the parallel points and some of the shoulder points.

A curve segment $\mathcal{C}(t) = (x(t),\, y(t)),\ \ a \le t \le b$, is called *normal* if it has a finite number of critical points. It is clear that a rational parametric curve is always normal. The curve segment $\mathcal{C}(t) = (\sin(1/(2t-1)), \cos(1/(2t-1))),\ 0 \le t \le 1$, is not normal since it has an infinite number of flexes near $t = 1/2$. We always assume that $\mathcal{C}(t)$ is normal throughout this paper.

The following algorithm illustrates how to divide a curve segment into triangle convex segments. It computes all the critical points, at which the tangents of the curve will change its direction in the parameter increasing order.

Algorithm 2. The input is a normal curve segment $\mathcal{C}(t) = (x(t),\, y(t)),\ a \le t \le b$. The outputs are $t_i \in \mathcal{R}, 0 \le i \le n$ and $T_{i-}, T_{i+} \in \mathcal{R}^2,\ 0 \le i \le n-1$, such that $t_0 = a < t_1 < \cdots < t_n = b$ and T_{i-} and T_{i+} are the left and right tangent directions of $S[\mathcal{C}(t_i),\, \mathcal{C}(t_{i+1})]$, which is triangle convex for $i = 0, \cdots, n-1$.

1. Find the cusp points and the flexes by solving the following univariate equations: $x'(t) = y'(t) = 0$ and $x'(t)y''(t) - x''(t)y'(t) = 0$. Let the solutions be $s_i,\ i = 1, \cdots, l-1$. We assume that $s_0 = a < s_1 < \cdots < s_l = b$.
2. For $i = 0, \cdots, l-1$, find the parallel points from Eq. 4 in each interval $[s_i,\, s_{i+1}]$. Let the corresponding t to the parallel points be $r_{i_j},\ \ i = 0, \cdots, l-1,\ j = 1, \cdots, m_i$ and take $r_{i_0} = s_i$.
3. For $r_{i_{j-1}} < t < r_{i_j}$, get the unique shoulder point $\mathcal{C}(u_{i_j})$ from Eq. 5 on $S[\mathcal{C}(r_{i_{j-1}}),\, \mathcal{C}(r_{i_j})]$ if it exists.
4. Rearrange s_i, r_{i_j} and u_{i_j} in an ascending order and rename them as $t_i,\ \ i = 0, \cdots, n$.
5. Find the left and right tangent directions T_{i-} and T_{i+} for each segment $S[\mathcal{C}(t_i),\, \mathcal{C}(t_{i+1})], i = 0, \cdots, n-1$.

In steps 1 and 2 of Algorithm 2, we need to find all the solutions of a univariate equation $P(t) = 0$ in a given interval. If $P(t)$ is a polynomial in t, the method in [14] is applied to solve the equation, which can find all the real solutions of polynomial equations with degrees up to fifty within seconds. An alternative method is also referred to [28] of quadratical convergence. For a rational curve, its flexes and cusp points can also be identified using the algorithm provided in [17]. If $P(t)$ is a general function, the global optimization methods, e.g., [3] can be applied to find the minimal values for $P(t)^2$. If $P(t)^2$ achieves a minimal value at t_0 and $P(t_0) = 0$, then t_0 is a solution. For univariate equations, our experiment results show that these approaches are very efficient.

Figure 5 illustrates the resulted segments for some curve segments divided by the critical points: (1) cusp points, (2) flexes, (3) parallel points and (4) shoulder points.

We have the following theorem for the triangle convexity of the resulted curve segments from the curve segmentation Algorithm 2:

Theorem 3. *In Algorithm 2, $S[\mathcal{C}(t_i),\, \mathcal{C}(t_{i+1})], i = 0, \cdots, n-1$, are triangle convex.*

Proof. Without loss of generality, we may assume that the segment is $S[\mathcal{C}(t_0),\, \mathcal{C}(t_1)] = S[P_0,\, P_2]$ (S for short) with $P_i = (x_i,\, y_i),\ \ i = 0, 2$. We first make a rotation such that the x-axis is parallel to P_0P_2. Figure 6 shows all the possible forms of S. Since there exist no singular points or flexes in S and the sweeping angle of the tangent line from

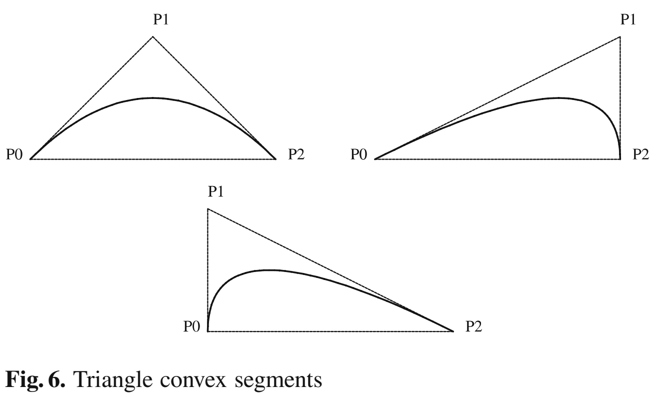


Fig. 6. Triangle convex segments

point P_0 to point P_2 is less than π, the slope $k(t)$ of S must be monotonic. We may assume that S is above line P_0P_2 and in this case $k(t)$ is decreasing. According to convex theory [6], a curve segment satisfying these conditions forms a convex region with P_0P_2. We need further to show that S is inside the control triangle. For an arbitrary point $P = (x, y) \neq (x_0, y_0)$ in S, there must exist a $\bar{t} > t_0$ such that point $(x(t), y(t))$ has the maximal distance to P_0P. At this point, we have $k(\bar{t}) = \frac{y-y_0}{x-x_0}$. On the other hand, there exists a point $(x, \bar{y})$ in the left tangent line of S such that $k(t_0) = \frac{\bar{y}-y_0}{x-x_0}$. Then

$$\frac{y-y_0}{x-x_0} = k(\bar{t}) < k(t_0) = \frac{\bar{y}-y_0}{x-x_0}.$$

We have $y < \bar{y}$. Then the point $P = (x, y)$ lies below the point $(x, \bar{y})$, a point in the left tangent line of S. In a similar way, we have that all the points in S lie below the right tangent line of S. Hence S is inside the control triangle. □

4 Shoulder point approximation for triangle convex segment

With Algorithm 2 in Sect. 3, we can divide a parametric segment into triangle convex segment. In this section, we will show how to approximate a triangle convex segment $\mathcal{C}(t) = (x(t), y(t)) = S[P_0, T_0, P_2, T_2]$.

Let P_1 be the intersection point of the left tangent line and the right tangent line. Then the family of rational quadratic curves $P(\omega, t)$ interpolating points P_0, P_2 with the tangent directions T_0, T_2 at P_0, P_2 can be represented as follows:

$$P(\omega, t) = \frac{P_0\phi_0(t) + \omega P_1\phi_1(t) + P_2\phi_2(t)}{\phi_0(t) + \omega\phi_1(t) + \phi_2(t)}, \quad 0 \le t \le 1, \qquad (6)$$

where the weight $\omega > 0$.

Suppose that the solid curve in Fig. 7 is the curve $\mathcal{C}(t)$ to be approximated and the dotted curves are the quadratic curve family $P(\omega, t)$. A proper value must be selected for ω so that we has an optimal approximation to $\mathcal{C}(t)$. In previous work, the free parameters are mainly determined under some interpolation constraints [5, 9, 20]. Since these methods are mainly based on the local properties of the approximated curve, they do not give good approximations in some special cases even if a high order of approximation can be achieved, as pointed out in the introduction section. If considering global properties, the selection of the weight ω usually leads to some optimization problems similar to the following:

$$\min_{\omega}(s(\mathcal{C}(t), P(\omega, t))), \quad \min_{\omega}(\max_{t}(d^2(\omega, t))), \qquad (7)$$

where $s(\mathcal{C}(t), P(\omega, t))$ is the area bounded by curves $\mathcal{C}(t)$ and $P(\omega, t)$; $d(\omega, t)$ is the distance function between $\mathcal{C}(t)$ and $P(\omega, t)$ expressed in various forms [1, 8, 22]. However all these expressions involve complicated computations and are impractical. For these reasons, others try to give a bound for the global error analysis in Eq. 7 so that the optimization problems can be simplified [1, 10].

In this section, we propose an approximation method for triangle convex segments, called *shoulder point approximation*. For a triangle convex segment, the shoulder point has certain global properties. That is, the shoulder point of a triangle convex segment has the maximum distance to the line determined by its two endpoints. We will push the shoulder points of the approximated curve and the approximate quadratic curve Eq. 6 as near as possible. This gives an optimization method which is fast and has a certain global property.

Before proposing the shoulder point approximation method, we will first show how to estimate the error between the approximated curve and its approximations. Suppose the implicit form of the quadratic curve Eq. 6 is $f(x, y) = 0$, which can be easily obtained from Eq. 3. The distance from $\mathcal{C}(t)$ to an implicitly defined function $f(x, y) = 0$ can be approximated by the following *approximation error function* [8]:

$$e(t) = \frac{f(x(t), y(t))}{[(f_x(x(t), y(t))^2 + f_y(x(t), y(t))^2)^2]^{\frac{1}{2}}}. \qquad (8)$$

The *approximation error* can then be set as the following optimization problem:

$$e = \max_{a \le t \le b}(e(t)). \qquad (9)$$

The optimization problem could be solved with existing methods [3]. In practice, we sample t as $t_i = \frac{i}{n}$, $i = 0, \cdots, n$, for a proper value of n, say $n = 20$, and set the approximation error as $\max(|e(t_i)|)$.

Let $\mathcal{C}(t) = (x(t), y(t))$, $a \le t \le b$, be a triangle convex curve segment and δ a small positive number. By a δ-approximation for $\mathcal{C}(t)$, we mean a sequence of numbers $a = t_0 < t_1 < \cdots < t_m = b$, and quadratic curve segments $P_i(t)$, $i = 1, \cdots, m$, such that:

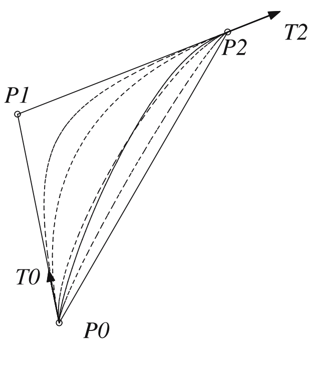


Fig. 7. Approximate curve family

– $P_i(t), i = 1, \cdots, m$, pass through $\mathcal{C}(t_{i-1}), \mathcal{C}(t_i)$ and have the same left and right tangent lines with $\mathcal{C}(t)$ at these points.
– The error between $\mathcal{C}(t)$ and $P_i(t)$ computed with Eq. 9 is less than δ in the interval (t_{i-1}, t_i).

The following algorithm gives a δ-approximation for a triangle convex segment via our shoulder point approximation method.

Algorithm 4. The input is a triangle convex curve segment $\mathcal{C}(t) = (x(t), y(t)) = S[P_0, T_0, P_2, T_2]$, $a \le t \le b$ and a small positive number δ. The output is a δ-approximation for $\mathcal{C}(t)$.

1. According to the interpolation requirements at the endpoints of $P(\omega, t)$, set $P(\omega, t)$ as Eq. 6.
2. Compute the shoulder point $M = (M_x, M_y)$ of $\mathcal{C}(t)$ from Eq. 5.
3. Let the shoulder point of $P(\omega, t)$ be $S(\omega)$, as expressed in Eq. 2. We will determine a specific value ω_0 for ω such that $S(\omega_0)$ has a minimum distance to the shoulder point M of $\mathcal{C}(t)$. Let $P_i = (x_i, y_i)$, $i = 0, 1, 2$. We have

$$S(\omega) = (S_x, S_y) = \left(\frac{x_0 + 2\omega x_1 + x_2}{2(1+\omega)}, \frac{y_0 + 2\omega y_1 + y_2}{2(1+\omega)} \right).$$

Solving the following equation, $\frac{\partial d^2(M,S)}{\partial \omega} = 0$, where $d^2(M, S) = (M_x - S_x)^2 + (M_y - S_y)^2$, we get

$$\omega_0 = \frac{1}{2} \cdot \frac{(x_0 + x_2 - 2M_x) + \alpha(y_0 + y_2 - 2M_y)}{(M_x - x_1) + \alpha(M_y - y_1)}$$

for $\alpha = \frac{y_0 + y_2 - 2y_1}{x_0 + x_2 - 2x_1}$.

4. Compute the approximation error $\bar{\delta}$ with Eq. 9 between $P(\omega_0, t)$ and $\mathcal{C}(t)$. $\bar{\delta} < \delta$, end this procedure. Otherwise, divide the segment into two parts at the shoulder point M and repeat the approximation method until the approximation error is less than δ.

The algorithm is ensured to be terminable for any small positive number δ from the theorem below.

Theorem 5. *With the Algorithm 4, the approximation error is convergent to zero. More precisely, let s be the area of the control triangle for the curve segment. After k recursive subdivisions, the distance between the approximate curve and the given curve is less than* $\sqrt{s}/2^k$.

Proof. Let $P(t)$ be the approximate curve for $\mathcal{C}(t)$ after one step of approximation. Then $P(t)$ and $\mathcal{C}(t)$ are contained in triangle $P_0P_1P_2$ (Fig. 8). After another step of segmentation, the curves are contained in triangles P_0Q_1M and P_2Q_2M, respectively. Let S_1 and S_2 points on P_0P_2 such that $MS_1 \parallel P_0Q_1$ and $MS_2 \parallel P_2Q_2$, s_0, s_1, s_2 and s the areas of triangles MS_1S_2, P_0Q_1M, P_2Q_2M and $P_0P_1P_2$, respectively. Then we have $(s_1 + s_2)/s_0 = Q_1Q_2/S_1S_2$

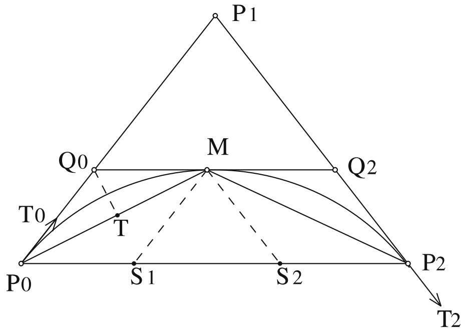


Fig. 8. Error control

and $s_0/s = (S_1S_2/P_0P_2)^2$. As a consequence,

$$\frac{s_0 + s_1}{s} = \frac{Q_1Q_2 \cdot S_1S_2}{P_0P_2^2} \le \frac{(Q_1Q_2 + S_1S_2)^2}{4P_0P_2^2} = \frac{1}{4}. \quad (10)$$

Due to the segmentation procedure, the angles $P_1P_0P_2$ and $P_1P_2P_0$ must be acute angles and hence the angles P_0Q_1M and P_2Q_2M must be obtuse angles. Let the heights of the triangles P_0Q_1M, P_2Q_2M corresponding to P_0M, P_2M be h_{10} and h_{11}, respectively. Let Q_1T be the altitude of triangle Q_1P_0M. We have $Q_1T^2 < P_0T \cdot TM \le (P_0T + TM)^2/4 = P_0M^2/4$. Then $h_{10} < P_0M/2$ and $h_{11} < P_2M/2$. From Eq. 10, we have that

$$h_{10}^2 + h_{11}^2 < h_{10} \cdot \frac{P_0M}{2} + h_{11} \cdot \frac{P_2M}{2} = s_0 + s_2 \le \frac{s}{4}.$$

In particular, we have $h_{10}^2 \le s/4$ and $h_{11}^2 \le s/4$. Repeat the process repeatedly. It is easy to show that after k steps of subdivisions, we have $h_{k0}^2 \le s/2^{2k}$ and $h_{k1}^2 \le s/2^{2k}$. □

Note that even for a large s, the error bound $\sqrt{s}/2^k$ will approach to zero very fast. This feature shows the global characteristic of the method.

Take for example the curve segment in Fig. 9 expressed as follows [8]:

$$\begin{aligned} \mathcal{C}_2(t) = (&t^6 + t^5 - 2t^3 + 3t^2 + 12t, \\ &t^6 - t^5 + t^4 - 4t^3 - 2t^2 + 24t), \end{aligned}$$

where $-1 \le t \le 1$.

In this example, $\mathcal{C}_2(t)$, $-1 \le t \le 1$, is first approximated by a single segment s_0. Then $\mathcal{C}_2(t)$ is divided into two segments at point $\mathcal{C}_2(.220)$ and the resulted segments are approximated by s_{00} and s_{01}, respectively. Since the approximation error for s_{00} is not under the error bound, $\mathcal{C}(t)$, $-1 \le t \le 0.220$ is further divided at point $\mathcal{C}_2(-.344)$ with corresponding approximations s_{000}, s_{001}. We compare respectively $\mathcal{C}_2(t)$, $-1 \le t \le 1$, with the approximate spline (s_0), (s_{00}, s_{01}), $(s_{000}, s_{001}, s_{01})$ in Fig. 9. The plots of the corresponding approximation error functions defined in Eq. 8 are also shown in Fig. 9.

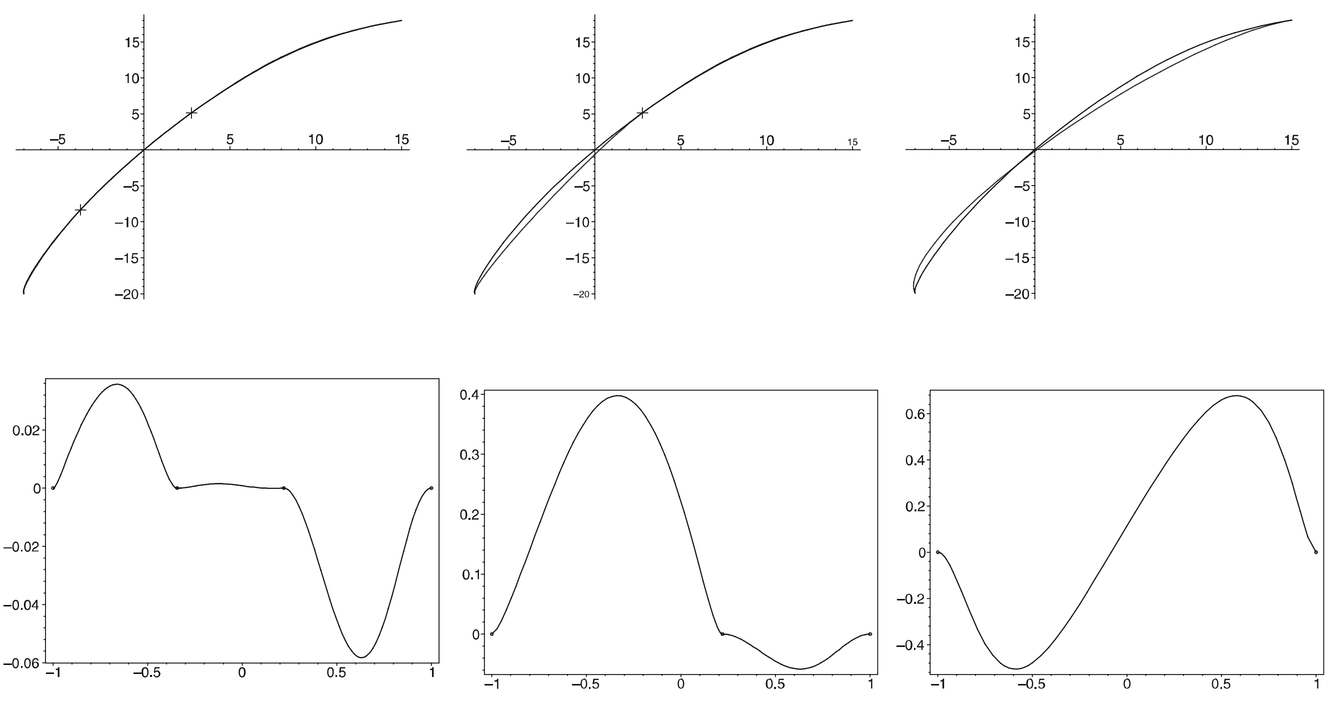


Fig. 9. Approximate spline for $\mathcal{C}_2(t)$

5 The algorithm and experimental results

With Algorithm 2 to divide a parametric curve into a triangle convex curve segment, which can then be approximated using Algorithm 4 with rational quadratic segments in the form of Eq. 1, we can now give the main approximation algorithm for a normal parametric curve segment $\mathcal{C}(t),\ a \leq t \leq b$ together with its corresponding approximate implicitizations.

Algorithm 6. The input is a normal curve segment $\mathcal{C}(t) = (x(t), y(t)),\ a \leq t \leq b$, and a small positive number δ. The outputs are a quadratic B-spline curve $B(t)$ such that the approximation error between $B(t)$ and $\mathcal{C}(t)$ is less than δ and the corresponding approximate implicitization spline for $\mathcal{C}(t)$.

1. Divide the curve into triangle convex segments with Algorithm 2. Let the parametric values corresponding to the critical points be $t_i,\ i = 0, \cdots, n+1$ (a, b are also included), and take the left and right tangent directions for each resulted segment as T_{i-} and $T_{i+},\ i = 0, \cdots, n$.
2. Construct a δ-approximation for $\mathcal{C}(t)$ on each interval (t_i, t_{i+1}) using Algorithm 4.
3. Implicitize each resulted quadratic segment from the step above as Eq. 3.
4. Convert the resulted rational quadratic Bézier spline curve into a rational quadratic B-spline with a proper knot selection just as the method proposed in [4, 19].

Theorem 7. *With Algorithm 6, we obtain a piecewise C^1 continuous approximate curve which keeps the convexity and the cusp (sharp) points of the approximated parametric curve.*

Proof. It can be first seen that the piecewise quadratic approximate splines is G^1 continuous from the fact that the quadratic curves have the same tangent directions with the original curve. The C^1 continuity is ensured from the conversion from piecewise Bézier spline into B-spline with a proper knot selection [4, 19]. Since the quadratic curves have no cusp points, we do not introduce new cusp points. On the other hand, for each segment with the cusp point as endpoints, since the original curve is normal, we may define the tangent directions at the cusp points and the quadratic curve segments have the same tangent directions at these cusp points. Therefore, the cusp points of the original curve are kept (see examples $\mathcal{C}_4(t)$ in Sect. 5). Furthermore, the curve is divided into triangle convex segments and the quadratic segments are convex with no flexes, we also keep the convexity of the curve. $\square$

The method reported here is implemented in Visual C++. The following experiments show that the method is quite efficient in terms of computation time and the number of segments.

Consider $\mathcal{C}_0(t)$ and $\mathcal{C}_1(t)$ defined in the introduction, $\mathcal{C}_3(t)$ and $\mathcal{C}_4(t)$ from [8], $\mathcal{C}_5(t)$ from [15] and $\mathcal{C}_6(t)$ introduced by ourselves as follows:

$$\mathcal{C}_3(t) = (3t^6 + t^5 - 2t^4 + 38t^3 - 5t^2 - 14t,$$
$$t^6 - 12t^5 - 2t^4 + 2t^3 - 7t^2 + 13t),$$
$$\mathcal{C}_4(t) = \left(\frac{5t^5 - 16t^4 + 10t^3 + 4t^2}{0.1t^3 + 0.1t^2 - 2t + 12.5}, \right.$$
$$\left. \frac{t^5 + t^4 + 2t^3 - 16t^2}{0.1t^3 + 0.1t^2 - 2t + 12.5} \right),$$
$$\mathcal{C}_5(t) = (92 - 93t - 8t^2 + 45t^3 - 59t^4 + 57t^5 +$$
$$63t^6 + 49t^7,\ -12 - 50t - 61t^2 + 99t^3 - 5t^4$$
$$+ 54t^5 + 66t^6 + 77t^7 - 62t^8 + 43t^9),$$
$$\mathcal{C}_6(t) = (\sin(2t) + \ln(5t^4 + 2) + 3t^2,$$
$$3e^{t^2-1} + \cos(t/5) + 2t^7).$$

The parameters for curves $\mathcal{C}_0(t)$, $\mathcal{C}_1(t)$, $\mathcal{C}_3(t)$, $\mathcal{C}_4(t)$, $\mathcal{C}_5(t)$ and $\mathcal{C}_6(t)$ take values in $[0, 1.8]$, $[0, 1]$, $[-1, 1]$, $[-1, 2.25]$, $[-0.8, 0.8]$ and $[-1, 1]$. The figures of the approximate splines and the plots of their corresponding error functions are shown in Figs. 10, 11, 12, 13, 14 and 15. The times needed to compute the approximate spline are 0.28, 0.34, 0.55, 2.09, 0.56 and 3.09 seconds, respectively, in a PC compatible with a 1.6G CPU.

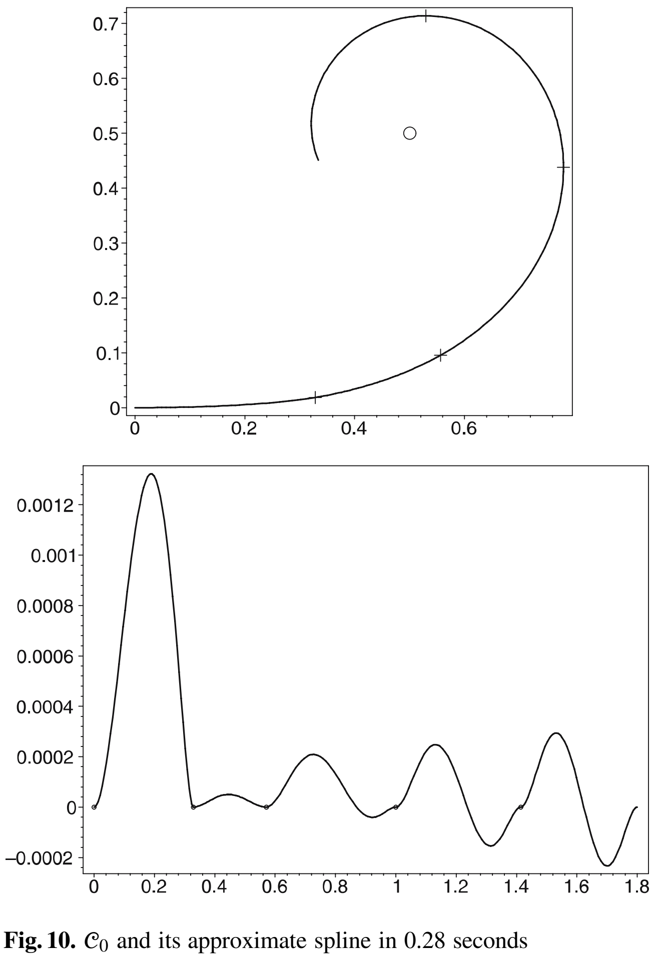


Fig. 10. $\mathcal{C}_0$ and its approximate spline in 0.28 seconds

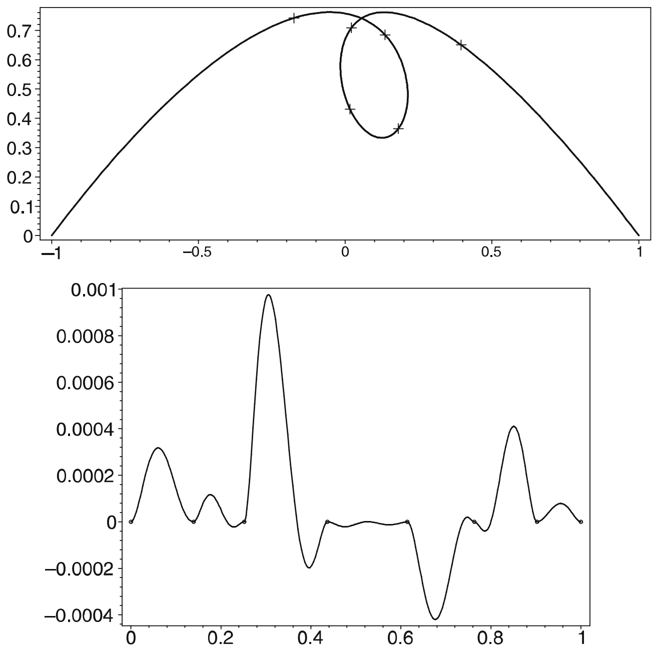


Fig. 11. $\mathcal{C}_1$ and its approximate spline in 0.34 seconds

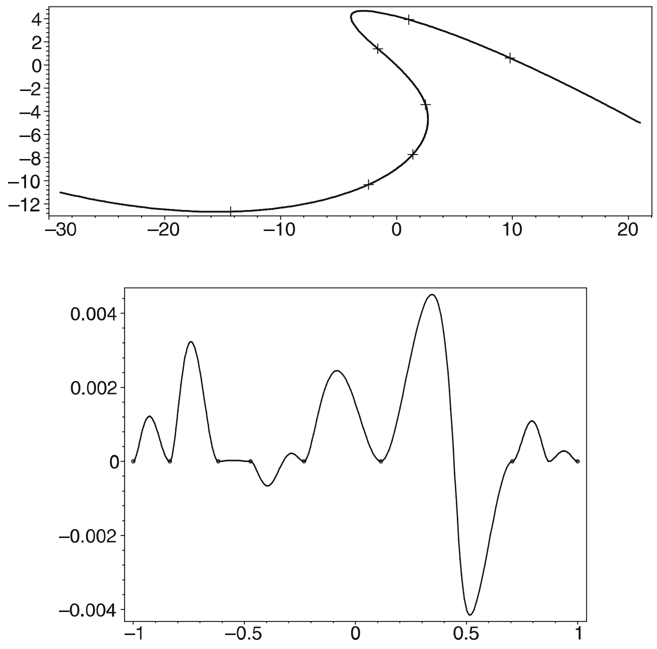


Fig. 12. $\mathcal{C}_3$ and its approximate spline in 0.55 seconds

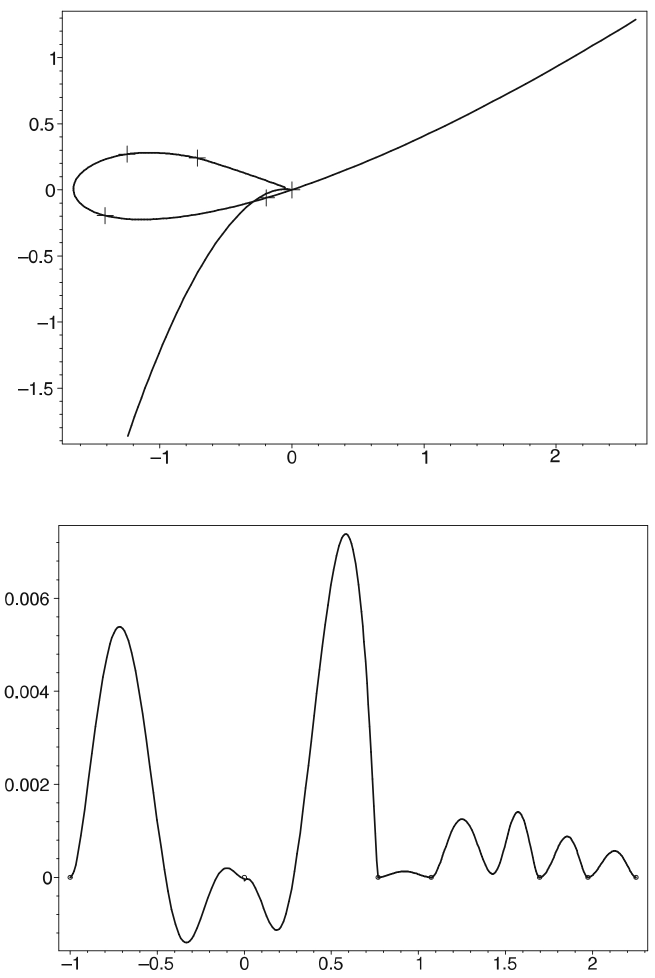


Fig. 13. $\mathcal{C}_4$ and its approximate spline in 2.09 seconds

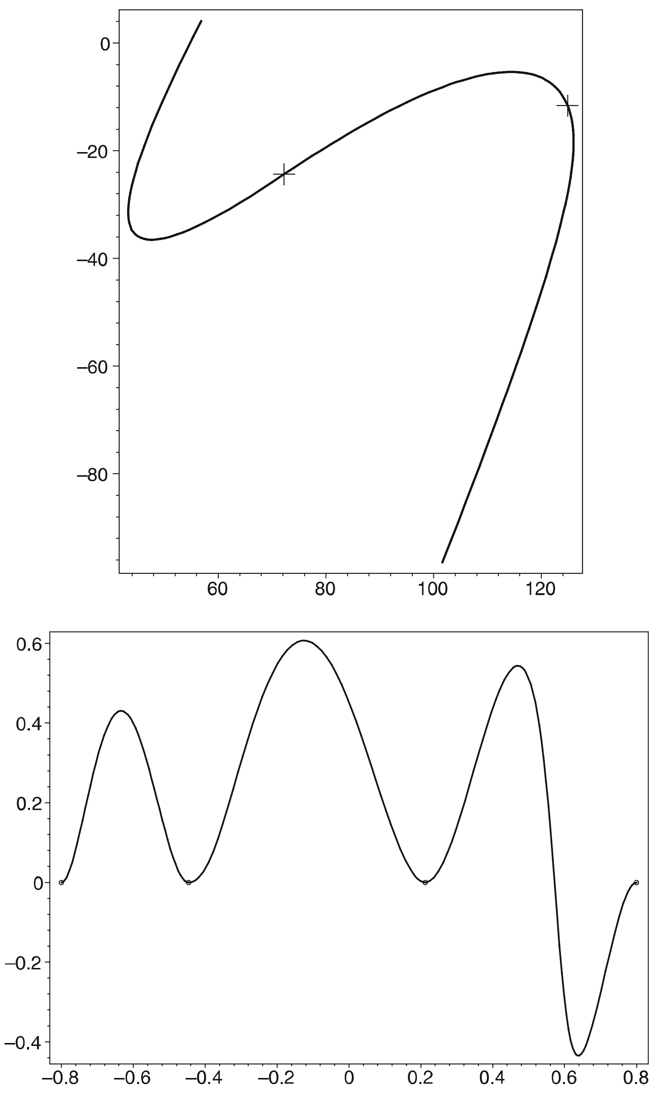


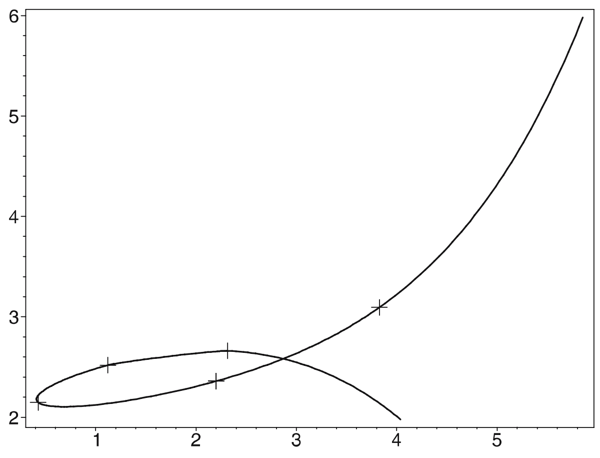
Fig. 14. $\mathcal{C}_5$ and its approximate spline in 0.56 seconds

We list the exact implicit forms of the curves $\mathcal{C}_i(t)$ as $f_i(x, y) = 0$, $i = 3, 4, 5$ in the following:

$$\begin{aligned}
f_3(x, y) = {} & x^6 - 18x^5y - 9960591x^5 + 135x^4y^2 \\
& - 1997264x^4y - 619679130x^4 - 540x^3y^3 \\
& + 3580432x^3y^2 + 350654542x^3y - 26526524706x^3 \\
& + 1215x^2y^4 - 8663560x^2y^3 - 429981958x^2y^2 \\
& - 135962440462x^2y - 943770338935x^2 \\
& - 1458xy^5 - 563721xy^4 - 4439136276xy^3 \\
& - 215654524172xy^2 - 3897453715476xy \\
& + 14764617650081x + 729y^6 - 532873y^5 \\
& - 297979610y^4 - 425140748152y^3 \\
& - 2007289936389y^2 + 15900357469318y.
\end{aligned}$$

$$\begin{aligned}
f_4(x, y) = {} & 3991794925x^5 - 4172700895x^4y \\
& + 6971917680x^4 + 2091059167x^3y^2 \\
& - 5264748720x^3y + 615988800x^3 \\
& - 543605072x^2y^3 - 39483415440x^2y^2 \\
& + 3079944000x^2y + 68106998xy^4 \\
& + 1194822720xy^3 - 4927910400xy^2 \\
& - 3206723y^5 + 350832510y^4 - 29567462400y^3.
\end{aligned}$$

$$\begin{aligned}
f_5(x, y) = {} & 271818611107x^9 - 1884921382721797x^8 \\
& + 4684978103434262x^7y \\
& + 6331914982244916011x^7 \\
& - 1915604648060992x^6y^2 \\
& - 18020143309088776518x^6y \\
& \cdots \\
& - 867257790753729562854841038447y^2 \\
& - 2825262401200651738516822470811 9y \\
& - 4003865370712281432329387032 29756.
\end{aligned}$$



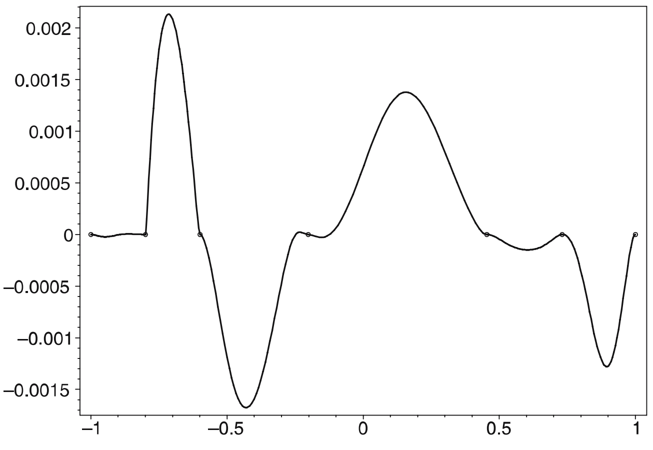


Fig. 15. $\mathcal{C}_6$ and its approximate spline in 3.09 seconds

The term $f_5(x, y)$ is a polynomial of degree 9 having 41 monomials with large numbers as coefficients. $\mathcal{C}_0(t)$ and $\mathcal{C}_6(t)$ have no exact implicit form. As a comparison, we also list the implicit forms of the three approximate segments for $\mathcal{C}_5(t)$ as follows:

$$\begin{aligned} f_{50} &= 13.443 - .221x + .558y + .001x^2 \\ &\quad - .004yx + .001y^2, \\ f_{51} &= 27031.010 - 448.412x + 440.475y + 1.888x^2 \\ &\quad - 3.071yx + 1.420y^2, \\ f_{52} &= 21.581 - .600x + .456y + .004x^2 \\ &\quad - .007yx + .003y^2. \end{aligned}$$

6 Conclusion

We propose an algorithm to construct rational quadratic B-spline approximation for a plane curve in parametric form and explore its applications in approximate implicitization. After the computation of the critical points, the algorithm mainly involves the solution of non-linear univariate equations. With this algorithm, we keep the convexity and the cusp points of the parametric curve with simple computation.

Our future work is to further explore the curve segmentation to intrinsically decompose the approximated curve in an optimal way for conic or cubic segments approximation, in combination with its curvature variation. The extension of the algorithm for an approximate parametrization of an implicitly defined algebraic curve can be found in [12]. It is interesting to see whether the method can be extended to approximate 3D surfaces.

We wish to express our thanks to referees of this paper for their valuable comments and suggestions.

References

1. Ahn, Y.: Conic approximation of plane curves. Comput. Aided Des. **33**(12), 867–872 (2001)
2. Bajaj, C., Xu, G.: Piecewise rational approximation of real algebraic curves. J. Comput. Math. **15**(1), 55–71 (1997)
3. Bazarra, M., Sherali, H., Shetty, C.: Non-linear Programming: Theory and Algorithms. Wiley, New York (1993)
4. Blomgren, R., Fuhr, R.: Algorithm to convert between rational b-spline and rational Bézier representation of curves and surfaces. Boeing Commercial Airplane Company, Renton, WA (16) (1982)
5. de Boor, C., Höllig, K., Sabin, M.: High accuracy geometric hermite interpolation. Comput. Aided Geom. Des. **4**(4), 269–278 (1987)
6. Chang, G., Sederberg, T.: Over and Over Again. The Mathematical Association of America, Washington DC (1998)
7. Cho, W., Maekawa, T., Patrikalakis, N.: Topologically reliable approximation of composite bezier curves. Comput. Aided Geom. Des. **13**(6), 497–520 (1996)
8. Chuang, J., Hoffmann, C.: On local implicit approximation and its application. ACM Trans. Graphics **8**(4), 298–324 (1989)
9. Degen, W.: High accurate rational approximation of parametric curves. Comput. Aided Geom. Des. **10**(3–4), 293–313 (1993)
10. Dokken, T.: Approximate implicitization. In: Lyche, T., Schumaker, L.L. (eds.) Mathematical Methods in CAGD. Vanderbilt University Press, Nashville (2001)
11. Farin, G.: Curvature continuity and offsets for piecewise conics. ACM Trans. Graphics **8**(2), 89–99 (1989)
12. Gao, X., Li, M.: Rational quadratic approximation to plane real algebraic curves. Comput. Aided Geom. Des. **21**(8), 805–828 (2004)
13. Hu, J., Pavlidis, T.: Function plotting using conic splines. IEEE Comput. Graphics Appl. **11**(1), 89–94 (1991)
14. Johnson, J.: Algorithms for polynomial real root isolation. In: Caviness, B.F., Johnson, J.R. (eds.) Quantifier Elimination and Cylindrical Algebraic Decomposition, pp. 269–299. Springer, Berlin Heidelberg New York (1998)

15. Kotsireas, I., Lau, E.: Implicitization of polynomial curves. In: Computer Mathematics. World Scientific, Singapore (2003)
16. Lee, E.: The rational Bézier representation for conics. In: Farin, G. (ed.) Geometric Modeling: Algorithm and New Trends, pp. 3–19. SIAM, Philadelphia (1985)
17. Li, Y., Cripps, R.: Identification of inflection points and cusps on rational curves. Comput. Aided Geom. Des. **14**(5), 491–497 (1997)
18. Montaudouin, Y., Tiller, W., Vold, H.: Application of power series in computational geometry. Comput. Aided Des. **18**(10), 93–108 (1986)
19. Park, H.: Choosing nodes and knots in closed b-spline curve interpolation to a point data. Comput. Aided Des. **33**(13), 967–974 (2001)
20. Piegl, L.: Interactive data interpolation by rational bézier curves. IEEE Comput. Graphics Appl. **7**(4), 45–58 (1987)
21. Pottmann, H.: Locally controllable conic splines with curvature continuity. ACM Trans. Graphics **10**(4), 366–377 (1991)
22. Pottmann, H., Leopoldseder, S., Hofer, M.: Approximation with active b-spline curves and surfaces. In: Coquillart, S., Shum, H.Y. (eds.) Pacific Graphics 2002 Proceedings. IEEE Computer Society, Los Alamitos, CA (2002)
23. Pratt, V.: Techniques for conic splines. ACM Trans. Graphics **19**(3), 151–159 (1985)
24. Quan, L.: Conic reconstruction and correspondence from two views. IEEE Trans. Patt. Analy. Mach. Intell. **18**(2), 151–160 (1996)
25. Sánchez-Reyes, J., Chacón, J.: Polynomial approximation to clothoids via s-power series. Comput. Aided Des. **35**(14), 1305–1313 (2003)
26. Sederberg, T., Zheng, J., Klimaszewski, K., Dokken, T.: Approximate implicitization using monoid curves and surfaces. Graphic. Models Images Process. **61**(4), 177–198 (1999)
27. Shalaby, M., Juttler, B., Schicho, J.: c^1 spline implicitization of planar curves. In: Automated Deduction in Geometry, pp. 161–177 (2002)
28. Sherbrooke, E., Patrikalakis, N.: Computation of the solution of non-linear polynomial systems. Comput. Aided Geom. Des. **10**(5), 379–405 (1993)
29. Wang, G., Sederberg, T., Chen, F.: On the convergence of polynomial approximation of rational functions. J. Approx. Theory **89**(3), 267–288 (1997)
30. Wang, W., Pottmann, H., Liu, Y.: Fitting b-spline curves to point clouds by squared distance minimization. ACM Trans. Graphics **25**(2), 214–238 (2006)
31. Yang, X.: Curve fitting and fairing using conic splines. Comput. Aided Des. **36**(5), 461–472 (2004)

MING LI is currently a research fellow in the School of Computer Science, Cardiff University, UK. He received his MSc degree from Jilin University in 2001, China and Ph.D. degree in mathematics from the Chinese Academy Sciences in 2004. His research interests include CAD, CAGD and algebraic geometry. His current work mainly involves curve/surface approximation and reverse engineering.

XIAO-SHAN GAO is a professor in the Institute of Systems Science, Chinese Academy of Sciences. His research interests include: automated reasoning, symbolic computation, intelligent CAD, CAGD, and robotics. He has published over one hundred research papers, two monographs and edited four books and conference proceedings (http://www.mmrc.iss.ac.cn/∼xgao).

SHANG-CHING CHOU currently a Professor at the Department of State University, received a Ph.D. at University of Texas at Austin in 1985. Since then he has been supported by NSF consecutively for his research in automated geometric reasoning.