Computational Visual Media
https://doi.org/10.1007/s41095-020-0186-4 Vol. 6, No. 4, December 2020, 417–430
Research Article

Computing knots by quadratic and cubic polynomial curves

Fan Zhang[1,2] (✉), **Jinjiang Li**[1,2], **Peiqiang Liu**[1,2], **and Hui Fan**[1,2]

Abstract A new method is presented to determine parameter values (knot) for data points for curve and surface generation. With four adjacent data points, a quadratic polynomial curve can be determined uniquely if the four points form a convex polygon. When the four data points do not form a convex polygon, a cubic polynomial curve with one degree of freedom is used to interpolate the four points, so that the interpolant has better shape, approximating the polygon formed by the four data points. The degree of freedom is determined by minimizing the cubic coefficient of the cubic polynomial curve. The advantages of the new method are, firstly, the knots computed have quadratic polynomial precision, i.e., if the data points are sampled from a quadratic polynomial curve, and the knots are used to construct a quadratic polynomial, it reproduces the original quadratic curve. Secondly, the new method is affine invariant, which is significant, as most parameterization methods do not have this property. Thirdly, it computes knots using a local method. Experiments show that curves constructed using knots computed by the new method have better interpolation precision than for existing methods.

Keywords knot; interpolation; polynomial curve; affine invariant

1 Introduction

1.1 Problem

A fundamental problem in the fields of computer-aided design, engineering, scientific computing, and computer graphics is the construction of curves and surfaces with high precision and smoothness. They require different attributes for different applications [1–6]. To meet these requirements, good interpolation techniques and parameterization methods are needed. For scientific computation and engineering application, constructing curves and surfaces with high polynomial accuracy is desirable. This paper focuses on how to determine the parameter values, or knots, for a given set of points with high precision.

1.2 Previous work

For a given set of data points to be interpolated, $P_i = (x_i, y_i)$, $i = 1, \ldots, n$, the aim of parameterization is to assign a parameter or knot value t_i, $t_1 < t_i < t_n$, for each P_i. The interpolated curve can be seen as the path of a particle through space, while the parameter t can be regarded as time, so the parameterization gives the location of the particle at each moment of time. For the same set of data, even with the same interpolation methods, constructing curves with different parameterizations will result in a different interpolant. The choice of parameterization method will have a noticeable effect on the interpolated curve. Uniform parameterization is only suitable for cases when the intervals between consecutive data points are equal. In applications, three non-uniform parameterization strategies are widely used: the chord length method [7], Foley's method [8], and the centripetal method [9]. The chord length method is a sound parameterized method, as parameter spacings reflect the chord lengths between consecutive data points. However, this interpolation only works well when the parametric curve is a straight line. The centripetal method assumes that, for a single arc, the centripetal force is proportional

1 School of Computer Science and Technology, Shandong Technology and Business University, Yantai 264005, China. E-mail: F. Zhang, zhangfan51@sina.com (✉); J. Li, lijinjiang@gmail.com.
2 Co-Innovation Center of Shandong Colleges and Universities: Future Intelligent Computing, Yantai 264005, China. E-mail: P. Liu, liupq@126.com; H. Fan, fanlinw@263.net.
Manuscript received: 2020-03-30; accepted: 2020-06-17

to the change in angle of the curve tangent vector from start to end of the arc. Foley's method is an adaptive chord parameterization method giving good planar parameterization results. But in terms of the approximation error, our experiments show that none of them can produce a satisfactory result.

The works by Lee [9] and Jeong et al. [10] provide improved approaches for curves and surfaces whose curvature changes greatly and is irregular. However, our experiments show that the Jeong et al.'s method generally results in more errors than Lee's method. In addition, out of the chord length method, Foley's method, and the centripetal method, only the last can assure no local self-intersection of the constructed curve [11]. Yuksel et al. [11] gave an analysis of these three methods, and showed that the curve produced by centripetal parameterization is most visually appropriate, but no mathematical explanation was given. Therefore, although the centripetal method is especially suitable for unevenly distributed data points, the resulting interpolation curve of these methods does not always capture all data characteristics. Fang and Hung [12] refined the interpolation results of the centripetal method to better capture wiggles, especially for interpolation of abruptly changing data. Lim [13] presented a new universal parameterization for B-spline interpolation, with better performance than existing parameterizations such as the ones in Refs. [7–9] by depending on the nature of B-spline basis functions, but experiments showed that this method could not improve the precision of the interpolation curves.

We note that the interpolation precision of the knot location methods previously mentioned is only linear, so if knot-set computed by the above methods is used to construct interpolation curve, the interpolating curve will not be a quadratic polynomial curve when the data points are sampled from a quadratic polynomial curve. If the data points are sampled from a non-linear curve, e.g., a quadratic or cubic polynomial curve, we would hope to reconstruct the underlying high-order curve. Zhang et al. [14] proposed a global method for choosing knots such that the interpolant constructed from the knots can exactly reproduce a quadratic polynomial curve. The approximation is better than that of linear precision methods in terms of error evaluation using the associated Taylor series. Based on this method, a local method for determining knots with quadratic precision was introduced in Ref. [15]. Even though this method employs a local computation, it has the ability to preserve quadratic precision. Hartley and Judd [16] discussed two ways of choosing knots: an iterative method and a simple formula. As the B-spline nodes were used as parameter values, it can achieve good shape and good parameterization. Martin [17] proposed a method of choosing knots through optimization for parametric cubic spline interpolation. In Ref. [18], the key concept was to generate a unique curve by minimizing its stress and stretching energies. An explicit function with high precision was constructed to compute the knots directly, which avoids solving a non-linear optimization problem. Unfortunately, this method is not invariant under affine transformation, as the knot was determined from only three consecutive points. The number of control points for constructing the curve plays an important role. To solve the problem that control points are redundant or inadequate, for the 3D case, Ref. [19] extended the planar case [20], and proposed an adaptive addition and removal process to refine the control points for the B-spline curve. Some articles also discuss the parameterization problems of spatial data points for other applications, while Refs. [21–24] construct parametric surfaces via parameterization.

Parameterization for curve and surface construction is still an open problem and has attracted considerable attention. Motivated by Ref. [25], Lü [7] identified a family of curves that can be parameterized by rational chord-length, and studied how the rational quartic and cubic curves could be applied to G^1 Hermite interpolation. Similarly, Bastl et al. [26, 27] extended chord length parameterization of rational curves to a family of RCL surfaces in any dimension. Tsuchie and Okamato [28] introduced a curvature continuous G^2 quadratic B-spline curve for fitting planar curves. The curve is constructed with non-uniform knots to ensure the G^2 condition, thereby reducing redundant segments in comparison to the use of uniform knots. To simplify the complicated optimization problem, Ref. [28] calculated the control points and adjusted the knot vector of the B-spline curve separately. Han [29] also discussed geometrically continuous splines in curve design, presenting a class of general quartic

splines for a non-uniform knot vector. The generated quartic spline curves have C^2 continuity with three local adjustable shape parameters, which greatly influence the shape of the spline curve. Bashir [30] presented the rational quadratic trigonometric Bézier curve with two shape parameters. Two segments of the objective curve can be joined with G^2 and C^2 continuity. Stepping outside the classical tensor product setting, Ref. [31] assigned a different parameter interval to each mesh edge, allowing interpolation of each section's polyline at parameter values that can prevent wiggling or other interpolation artifacts, to yield high-quality interpolating surfaces.

1.3 Proposed method

This paper provides a new method for computing knots. Its derivation is based on the assumption that the given set of data points have been sampled from a parametric curve that can be approximated well by piecewise quadratic or cubic polynomial curves. In particular, it assumes that each curve segment between four adjacent points can be approximated by a quadratic polynomial or a cubic polynomial. If the four adjacent consecutive data points form a convex polygon, the four data points are sufficient for determining a unique interpolating quadratic polynomial curve. Otherwise, a cubic polynomial curve with one degree of freedom is used to interpolate the four points; this degree of freedom is determined by minimizing the cubic coefficient of the cubic polynomial curve. This technique constructs quadratic and cubic polynomial curves consistently, in the sense that for a quadratic polynomial curve, its cubic coefficient is zero, while for a cubic polynomial curve, its cubic coefficient is as small as possible. Minimizing the cubic coefficient of the cubic polynomial curve ensures that the cubic polynomial curve approximates the polygon formed by the four data points well, and hence has excellent shape. As the knots are determined by the quadratic curve and the cubic curve, they can reflect the distribution of the data points well. When the quadratic and cubic polynomial functions are determined, computing the knot values for each data point is an easy task.

The new method has several advantages. Firstly, the knots computed have quadratic polynomial precision: if the data points are sampled from an underlying quadratic polynomial curve, and the knots are used to construct a quadratic polynomial interpolant, the resulting curve reproduces the underlying quadratic curve. Therefore, when used for curve construction, the resulting curve has higher precision than methods with linear precision. Secondly and importantly, the new method is affine invariant. Furthermore, our method is a local method, so it is easy to modify a curve interactively, making the curve design process efficient and flexible. Experiments show that curves constructed with knots computed by the new method have better interpolation precision than curves constructed using knots from existing methods. Experiments show that the approximation precision of our method is better than for methods in Refs. [7–9, 12, 15, 18, 27].

2 Basis of the new method

Let $P_i = (x_i, y_i), 1 \leqslant i \leqslant n$, be a given set of distinct data points. The goal is to compute a knot value t_i for each point P_i. When the knots are used to construct a parametric curve $P(t)$ interpolating all $P_i = (x_i, y_i)$ using an existing interpolation method, $P(t)$ should have quadratic polynomial precision as defined above.

The main idea of the new method is as follows. For each point P_i, we locally compute a knot t_i from consecutive data points. For the two sets of consecutive data points corresponding to P_i, $\{P_{i-2}, P_{i-1}, P_i, P_{i+1}\}$ and $\{P_{i-1}, P_i, P_{i+1}, P_{i+2}\}$, two respective curves $P_i(t)$ and $P_{i+1}(t)$ are constructed through the two sets. These curves are used to compute the knot t_i associated with P_i.

We compute the t_i to satisfy the following condition that, if the P_i are taken from a parametric quadratic polynomial, $P_i = P(u_i)$ where $P(u) = (x(u), y(u))$ is defined by

$$\begin{aligned} x(u) &= X_2u^2 + X_1u + X_0 \\ y(u) &= Y_2u^2 + Y_1u + Y_0 \end{aligned} \tag{1}$$

then

$$t_i = \alpha u_i + \beta, \quad 1 \leqslant i \leqslant n \tag{2}$$

for some constants α and β. This will ensure the quadratic precision requirement, since a linear transformation of knot values does not change the shape of a curve.

If the data points P_i, are taken from a quadratic polynomial defined by Eq. (1), any four consecutive data points $\{P_{i-2}, P_{i-1}, P_i, P_{i+1}\}$, $i = 3, 4, \cdots, n-1$ will uniquely determine a quadratic polynomial curve

$P_i(t)$ which is the same as $P(u)$ in Eq. (1), but possibly with a different parameterization. Let $t_j^i = \alpha_i u_j + \beta_i$ be the knots computed with respect to $P_i(t)$ passing through the four data points $\{P_{i-2}, P_{i-1}, P_i, P_{i+1}\}$. Let $t_j^{i+1} = \alpha_{i+1} u_j + \beta_{i+1}$ be the knots computed with respect to $P_{i+1}(t)$ passing through the four data points $\{P_{i-1}, P_i, P_{i+1}, P_{i+2}\}$. Although $P_i(t)$ and $P_{i+1}(t)$ are the same as the quadratic curve $P(u)$ in Eq. (1), they may have different parameterizations. Thus, we will have two sets of knot values t_j^i and t_j^{i+1} for the three data points P_j, $j = i-1, i, i+1$. Since the two sequences of knots t_j^i and t_j^{i+1}, $j = i-1, i, i+1$, are both linearly related to u_i, it is possible to use a linear mapping to match the two sequences. For each point P_i, as the knot t_i is locally computed using two curves $P_i(t)$ and $P_{i+1}(t)$, all t_i could have different parameterizations, so one needs to reparameterize t_i in a parameter space.

To develop a complete solution based on the idea above, we face two tasks. We must compute the local knot sequence t_j from two groups of four consecutive data points separately, and then we must merge all of these local knot sequences into a global knot sequence in a parameter space which has quadratic precision. These two steps will be explained in the following sections.

3 Computing knot s_i from consecutive data points

In this section, determining the knots of three consecutive points $\{P_{i-1}, P_i, P_{i+1}\}$ from their neighboring points is discussed in detail. For each set of four neighboring points $\{P_{i-2}, P_{i-1}, P_i, P_{i+1}\}$, a quadratic curve or cubic curve $Q_i(s)$ can be uniquely defined. For the next set of four points $\{P_{i-1}, P_i, P_{i+1}, P_{i+2}\}$, a quadratic curve or cubic curve also can be determined. The three points $\{P_{i-1}, P_i, P_{i+1}\}$ are shared by these two sequences. Note that each group of each three neighboring points is a participant in at least two adjacent sequences. The key is the combination of the two sequences of s_i and s_{i+1}.

3.1 Computing s_i from a quadratic polynomial

For five given consecutive points $P_j = (x_j, y_j)$, $j = i-2, i-1, i, i+1, i+2$, if P_{i-1}, P_i, and P_{i+1} are non-collinear, then using the following transformation:

$$\begin{aligned} x &= a_{11}(x - x_i) + a_{12}(y - y_i) \\ y &= a_{21}(x - x_i) + a_{22}(y - y_i) - h \end{aligned} \tag{3}$$

where

$$\begin{aligned} a_{11} &= \frac{y_{i-1} - 2y_i + y_{i+1}}{r} \\ a_{12} &= \frac{-x_{i-1} + 2x_i - x_{i+1}}{r} \\ a_{21} &= \frac{h(y_{i-1} - y_{i+1})}{r} \\ a_{22} &= \frac{h(x_{i+1} - x_{i-1})}{r} \\ r &= (x_{i+1} - x_i)(y_{i-1} - y_i) - (x_{i-1} - x_i)(y_{i+1} - y_i) \end{aligned} \tag{4}$$

the coordinates of P_{i-1}, P_i, and P_{i+1} can be transformed to $(-1, 0)$, $(0, -h)$, $(1, 0)$, as shown in Fig. 1. The quadratic polynomial $P_i(s)$ interpolating points $(-1, 0)$, $(0, -h)$, and $(1, 0)$ is as follows:

$$\begin{aligned} x &= \frac{(s - s_i)(1 - s)}{s_i} + \frac{s(s - s_i)}{1 - s_i} \\ y &= \frac{s(s-1)}{s_i(1 - s_i)} h \end{aligned} \tag{5}$$

where $0 < s_i < 1$ is a parameter to be determined, satisfying:

$$s_i = \frac{t_i - t_{i-1}}{t_{i+1} - t_{i-1}} \tag{6}$$

Theorem 1. In Eq. (5), the relationship between parameter s and point (x, y) is defined by

$$s = \frac{1 + x + y(1 - 2s_i)/h}{2} \tag{7}$$

Proof Eq. (5) may be rewritten as

$$\begin{aligned} x &= \frac{(s - s_i)(1 - s)(1 - s_i) + s(s - s_i)s_i}{s_i(1 - s_i)} \\ \frac{y}{h} &= \frac{s(s-1)}{s_i(1 - s_i)} \end{aligned} \tag{8}$$

We have

$$\frac{xh}{y} = \frac{(s - s_i)(1 - s)(1 - s_i) + s(s - s_i)s_i}{s(s-1)} \tag{9}$$

Now

$$\frac{xh}{y} = \frac{(s - s_i)(1 - s - s_i + 2s_i s) + s_i(1 - s_i)}{s(s-1)} - \frac{h}{y} \tag{10}$$

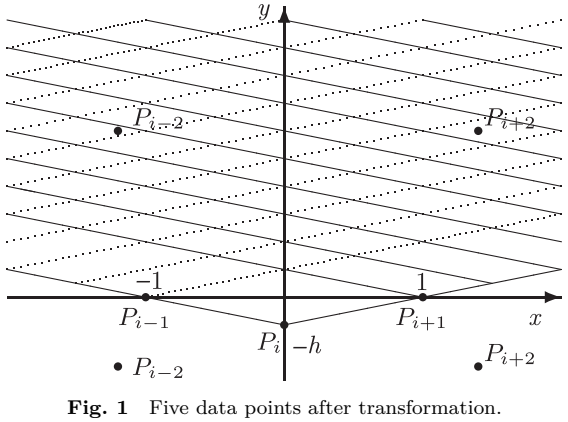

Fig. 1 Five data points after transformation.

By simple algebra, it follows from Eq. (10) that Eq. (7) holds.

Once the parameter s_i associated with (x_i, y_i) has been identified, the parameter value s at point (x, y) can be found using Eqs. (7) and (5) and simple algebra:

$$a(x,y)s_i^2 + b(x,y)s_i + c(x,y) = 0 \tag{11}$$

where

$$\begin{aligned} a(x,y) &= 4y(y+h) \\ b(x,y) &= -2y(\sigma(x,y) + \rho(x,y) + 2h) \\ c(x,y) &= \rho(x,y)\sigma(x,y) \end{aligned} \tag{12}$$

and

$$\begin{aligned} \rho(x,y) &= h(1+x) + y \\ \sigma(x,y) &= h(x-1) + y \end{aligned}$$

Parameter s_i is the solution of Eq. (11). The five points P_j, $j = i-2, i-1, i, i+1, i+2$, can be mapped via the affine mapping in Eq. (3) to the coordinates (x_{i-2}, y_{i-2}), $(0, 1)$, $(0, -h)$, $(1, 0)$, and (x_{i+2}, y_{i+2}), respectively, as shown in Fig. 1, so Eq. (11) is invariant under this affine mapping, and hence, the parameter s_i is invariant.

If the coordinate values of point (x_{i+2}, y_{i+2}) satisfy $y_{i+2} + h(x_{i+2} - 1) > 0$ and $y_{i+2} - h(x_{i+2} - 1) > 0$, i.e., point P_{i+2} is located in the area with solid lines, then the four points $\{P_{i-1}, P_i, P_{i+1}, P_{i+2}\}$ form a convex polygon, as shown in Fig. 1. When $(x, y) = (x_{i+2}, y_{i+2})$, the root of Eq. (11) is given by

$$s_i^r = \frac{-b(x_{i+2}, y_{i+2}) - \sqrt{G(x_{i+2}, y_{i+2})}}{2a(x_{i+2}, y_{i+2})} \tag{13}$$

where $G(x_k, y_k) = b(x_k, y_k)^2 - 4a(x_k, y_k)c(x_k, y_k)$, $k = i + 2$. Similarly, if the coordinate values of point (x_{i-2}, y_{i-2}) satisfy $y_{i-2} - h(x_{i-2} + 1) > 0$ and $y_{i-2} + h(x_{i-2} + 1) > 0$, i.e., point P_{i-2} is located in the area with dotted lines, as shown in Fig. 1. The root of Eq. (11) with $(x, y) = (x_{i-2}, y_{i-2})$ is given by

$$s_i^l = \frac{-b(x_{i-2}, y_{i-2}) + \sqrt{G(x_{i-2}, y_{i-2})}}{2a(x_{i-2}, y_{i-2})} \tag{14}$$

where $G(x_{i-2}, y_{i-2})$ defined as before. We may now state the following theorem:

Theorem 2. When $(x, y) = (x_{i+2}, y_{i+2})$, (x_{i-2}, y_{i-2}), the roots of Eq. (11) are s_i^r and s_i^l, respectively.

Proof For simplicity, (x, y) in Eq. (11) is set to $(0, y)$. Then $a(x, y)$, $b(x, y)$, and $c(x, y)$ in Eq. (12) become

$$\begin{aligned} a(x,y) &= 4y(y+h) \\ b(x,y) &= -4y(y+h) \\ c(x,y) &= (y+h)(y-h) \end{aligned}$$

Then, the two solutions of Eq. (11) are

$$s_i = \frac{y \pm \sqrt{hy}}{2y} \tag{15}$$

Substituting Eq. (15) into Eq. (7) gives

$$s_i = \frac{1}{2}\Big(1 - \frac{\pm\sqrt{hy}}{h}\Big) \tag{16}$$

For $(x, y) = (x_{i+2}, y_{i+2})$, s_i^r should satisfy $s_i^r > 1$: as $y > h$, it follows from Eq. (16) that s_i^r should be defined by Eq. (13). Similarly, for $(x, y) = (x_{i-2}, y_{i-2})$, $s_i^l < 0$ should be defined by Eq. (14).

3.2 Computing s_i from a cubic polynomial

In addition to above two cases, however, we must consider another one: when the coordinate values (x_j, y_j), $j = i-2, i+2$, fail to meet $y_{i+2} - h(x_{i+2} - 1) > 0$ and $y_{i-2} + h(x_{i-2} + 1) > 0$, i.e., points P_{i-2} and P_{i+2} are not located in the dotted line area and the solid line area, respectively, shown in Fig. 1. In this case, points $(-1, 0)$, $(0, -h)$, $(1, 0)$, and (x_j, y_j), $j = i - 2, i + 2$, do not form a convex polygon, so it is necessary to construct a cubic polynomial to interpolate the four consecutive points. Let the knots of points $P_j, j = i - 2, i + 2$, be s_j. Then

$$s_j = \frac{1 + x_j + y_j(1 - 2s_i)/h}{2} \tag{17}$$

The cubic polynomial interpolating the four consecutive points is

$$\begin{aligned} x &= -\frac{(s-s_i)(s-1)}{s_i} + \frac{s(s-s_i)}{1-s_i} + \frac{W(s)}{W(s_j)}X_j \\ y &= \frac{s(s-1)}{s_i(1-s_i)}h + \frac{W(s)}{W(s_j)}Y_j \end{aligned} \tag{18}$$

where

$$\begin{aligned} W(s) &= s(s-s_i)(s-1) \\ X_j &= x_j + \frac{(s_j-s_i)(s_j-1)}{s_i} - \frac{s_j(s_j-s_i)}{1-s_i} \\ Y_j &= y_j - \frac{s_j(s_j-1)}{s_i(1-s_i)}h \end{aligned} \tag{19}$$

Parameter s_i in Eq. (18) is determined by minimizing the cubic coefficient of Eq. (18), i.e., by minimizing the following objective function:

$$G(s_i) = \frac{X_j^2 + Y_j^2}{W(s_j)^2} \tag{20}$$

The definition of objective function Eq. (20) is reasonable. When $(-1, 0)$, $(0, -h)$, $(1, 0)$, and (x_j, y_j)

form a convex polygon, the cubic coefficient of the curve function is zero. While, when $(-1,0)$, $(0,-h)$, $(1,0)$, and (x_j,y_j) do not form a convex polygon, the cubic coefficient of the cubic curve should be as small as possible, enabling a slow and stable change of curve shape in both cases.

3.3 Computing s_i

When the five points $P_j=(x_j,y_j)$, $i-2\leqslant j\leqslant i+2$, are taken from the same quadratic curve, $s_i^l=s_i^r$. However, for data points in general positions (but still assumed to form a convex chain), these five points may not lie on the same underlying quadratic curve, so $s_i^l\neq s_i^r$. In this case, we must reconcile the two values to determine a knot s_i for P_i. An obvious choice would be to set $s_i=(s_i^l+s_i^r)/2$. In the following, a more elaborate, improved scheme to find s_i from s_i^l and s_i^r is proposed.

Reconsidering Eq. (11), let

$$h(x,y,s)=a(x,y)s^2+b(x,y)s+c(x,y)=0 \quad (21)$$

Then, s_i^l and s_i^r are roots of $h(x_{i-2},y_{i-2},s)$ and $h(x_{i+2},\,y_{i+2},s)$, respectively, as shown in Fig. 3. Let

$$s_i=s_i^l+r(s_i^r-s_i^l) \quad (22)$$

and

$$g(r)=h(x_{i-2},y_{i-2},s_i)^2+h(x_{i+2},y_{i+2},s_i)^2 \quad (23)$$

It is obvious that a good choice for s_i in Eq. (22) gives $g(r)$ a small value, as it follows from Eqs. (5)–(14) that small a $g(r)$ means that $P_i(s)$ in Eq. (5) well approximates the five data points (x_{i-2},y_{i-2}), $(0,1)$, $(0,0)$, $(1,0)$, and (x_{i+2},y_{i+2}). The shape of $g(r)$ is shown in Fig. 2. Our goal is to find a r^c satisfying

$$s_i^c=s_i^l+r^c(s_i^r-s_i^l)$$

so that $g(r^c)$ has the minimum value of $g(r)$, as shown in Fig. 2, and defined by

$$g(r^c)=h(x_{i-2},y_{i-2},s_i^c)^2+h(x_{i+2},y_{i+2},s_i^c)^2 \quad (24)$$

The value of r^c is determined by

$$\frac{\mathrm{d}g(r)}{\mathrm{d}r}=0 \quad (25)$$

Eqs. (21)–(23) show that Eq. (25) is a cubic equation, so it is easy to solve. Although we found no case in our experiments for which Eq. (25) has no root in the interval $(0,1)$, we handle such cases as follows: r_i^c is defined by

$$s_i^c=\begin{cases} s_i^l, & \text{if } g(0)<g(1)\\ s_i^r, & \text{otherwise}\end{cases} \quad (26)$$

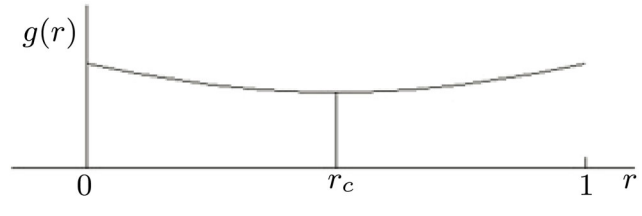

Fig. 2 Plot of Eq. (23).

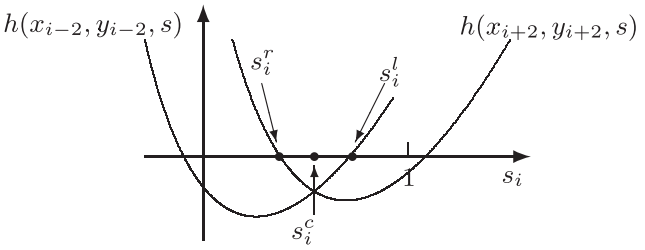

Fig. 3 Positions of s_i^l, s_i^c, and s_i^r.

Eqs. (13), (14), and (22) give three estimates for s_i: s_i^l, s_i^c, and s_i^r, respectively, as indicated in Fig. 3. We now compute s_i as a combination of s_i^l, s_i^c, and s_i^r. We first discuss how to define the weight $\omega(s_i)$ associated with knot s_i. Whichever of s_i^l, s_i^c and s_i^r is closest to 0.5 should have the biggest effect on s_i, so the weight $\omega(s_i^l)$ should be proportional to $s_i^l(1-s_i^l)$. Let

$$w(s)=\sqrt{h(x_{i-2},y_{i-2},s)^2+h(x_{i+2},y_{i+2},s)^2} \quad (27)$$

Obviously, for s_i^l, s_i^c, and s_i^r, the best case is that they satisfy $w(s_i^l)=w(s_i^c)=w(s_i^r)=0$. As in this case, each of s_i^l, s_i^c, and s_i^r makes the curves in Eq. (5) or (18) interpolate the five points $P_j=(x_j,y_j)$, $j=i-2,i-1,i,i+1,i+2$. Thus, the weight $\omega(s_i)$ associated with knot s_i should be inversely proportional to $w(s_i)$. Based on the discussion above, weight $\omega(s_i)$ is defined as

$$\omega(s_i)=\frac{s_i^2(1-s_i)^2}{w(s_i)} \quad (28)$$

If any of $w(s_i^l)$, $w(s_i^c)$, and $w(s_i^r)$ is zero, then s_i should be set equal to it. Otherwise, s_i is defined by a weighted combination of s_i^l, s_i^c, and s_i^r, given by

$$s_i=\frac{\omega(s_i^l)s_i^l+\omega(s_i^c)s_i^c+\omega(s_i^r)s_i^r}{\omega(s_i^l)+\omega(s_i^c)+\omega(s_i^r)} \quad (29)$$

3.4 Discussion

So far we have neglected the case where some three consecutive points are collinear. We now address it. When P_{i-1}, P_i, and P_{i+1} are on a straight line, we set

$$s_i=\frac{|P_{i-1}P_i|}{|P_{i-1}P_i|+|P_iP_{i+1}|} \quad (30)$$

This choice makes the quadratic polynomial through P_{i-1}, P_i, and P_{i+1} a straight line with constant first derivative. This is the most naturally defined curve for this case.

Finally, for the end data points, s_2 corresponding to $Q_2(s)$ is determined using the four points $P_j, j = 1,2,3,4$, and s_{n-1} corresponding to $Q_{n-1}(s)$ is determined using the points $P_j, j = n-3, n-2, n-1, n$.

4 Computing t_i with a local method

Based on the discussion above, with two sets of four data points $\{P_{j-1}, P_j, P_{j+1}, P_{j+2}\}$, $j = i-1, i$, one can construct two quadratic curves $P_i(s)$ and $P_{i+1}(s)$, with two knot intervals $1-s_i$ and s_{i+1} for P_i and P_{i+1}, respectively. For $P_i(s)$, the knot interval for P_{i-1} and P_{i+1} is set to $[0,1]$, while for $P_{i+1}(s)$, the knot interval for P_i and P_{i+2} is set to $[0,1]$. Hence $P_i(s)$ and $P_{i+1}(s)$ are defined on different parametric spaces. The reason is as follows. For $P_i(s)$, the knot interval for P_{i-1} and P_{i+1} is $[0,1]$, so from Eq. (6), the knot corresponding to P_{i+2} is

$$s_{i+2} = (t_{i+2} - t_{i-1})/(t_{i+1} - t_{i-1}) \tag{31}$$

Thus, for $P_i(s)$, the knots corresponding to P_i, P_{i+1}, and P_{i+2} are s_i, 1, and s_{i+2}, respectively. Since P_{i+2} could have any possible position, in general, s_i, 1, and s_{i+2} will not be 0, s_{i+1}, and 1 through the translation transformation defined in Eq. (6), i.e., the two sets $\{s_i, 1, s_{i+2}\}$ and $\{0, s_{i+1}, 1\}$ generally do not satisfy

$$\frac{1-s_i}{s_{i+2}-1} = \frac{s_{i+1}}{1-s_{i+1}}$$

In the following, we will use a normal form of a quadratic curve introduced in Ref. [15] to translate all $P_i(s)$, $1 < i < n$, into the same parameter space, then compute the knot interval for P_i and P_{i+1} by merging $1-s_i$ and s_{i+1}. All the knot intervals corresponding to data pairs P_{i-1} and P_i, $i = 2, \cdots, n-1$, are combined to form a consistent global knot sequence with respect to the same parameterization of a quadratic curve.

If the knots corresponding to P_{i-1}, P_i, and P_{i+1} are 0, s_i, and 1, respectively, then the quadratic polynomial $P_i(s)$ passing through these three data points can be written as

$$\begin{aligned} x_i(s) &= a_i s^2 + b_i s + x_{i-1} \\ y_i(s) &= d_i s^2 + e_i s + y_{i-1} \end{aligned} \tag{32}$$

where

$$\begin{aligned} a_i &= \frac{(x_{i-1}-x_i)(1-s_i) + (x_{i+1}-x_i)s_i}{s_i(1-s_i)} \\ b_i &= -\frac{a_i s_i^2 + x_{i-1} - x_i}{s_i} \\ d_i &= \frac{(y_{i-1}-y_i)(1-s_i) + (y_{i+1}-y_i)s_i}{s_i(1-s_i)} \\ e_i &= -\frac{d_i s_i^2 + y_{i-1} - y_i}{s_i} \end{aligned} \tag{33}$$

If $P_i(s)$ and $P_{i+1}(s)$ represent the same curve, they can be transformed into the normal form in Eq. (36), and they will have the same knot intervals between P_i and P_{i+1}.

Suppose that in Eq. (33), $a_i \neq 0$ or $d_i \neq 0$. By the following transformation Eq. (34):

$$\begin{aligned} \bar{x} &= x\cos\theta_i + y\sin\theta_i \\ \bar{y} &= -x\sin\theta_i + y\cos\theta_i \end{aligned} \tag{34}$$

where

$$\begin{aligned} \cos\theta_i &= \frac{a_i + d_i}{\sqrt{a_i^2 + d_i^2}} \\ \sin\theta_i &= \frac{d_i - a_i}{\sqrt{a_i^2 + d_i^2}} \end{aligned}$$

and a linear reparameterization Eq. (35):

$$t = \left(a_i^2 + d_i^2\right)^{\frac{1}{4}} s \tag{35}$$

$P_i(s)$ in Eq. (32) can be transformed into the following normal form:

$$\begin{aligned} \bar{x}_i(t) &= t^2 + \bar{b}_i t + \bar{d}_i \\ \bar{y}_i(t) &= t^2 + \bar{e}_i t + \bar{f}_i \end{aligned} \tag{36}$$

where

$$\begin{aligned} \bar{d}_i &= \cos\theta_i x_{i-1} + \sin\theta_i y_{i-1} \\ \bar{f}_i &= -\sin\theta_i x_{i-1} + \cos\theta_i y_{i-1} \\ \bar{b}_i &= \frac{\cos\theta_i b_i + \sin\theta_i e_i}{\sqrt{\cos\theta_i a_i + \sin\theta_i d_i}} \\ \bar{e}_i &= \frac{-\sin\theta_i b_i + \cos\theta_i e_i}{\sqrt{\cos\theta_i a_i + \sin\theta_i d_i}} \end{aligned} \tag{37}$$

Also, the properties of the above argument are invariant under such a normal form transformation of the quadratic polynomial.

When the quadratic curve $P_i(s)$ in Eq. (32) is transformed into the normal form in Eq. (36) by the reparameterization process in Eqs. (34) and (35), the knot intervals s_i and $1-s_i$ in Eq. (32) become Δ_{i-1}^i and Δ_i^i, respectively, defined by

$$\begin{aligned} \Delta_{i-1}^i &= \left(a_i^2 + d_i^2\right)^{\frac{1}{4}} s_i \\ \Delta_i^i &= \left(a_i^2 + d_i^2\right)^{\frac{1}{4}} (1 - s_i) \end{aligned} \tag{38}$$

where a_i and d_i are defined in Eq. (33).

By mapping each $P_i(s)$ into normal form, for each pair of consecutive points P_i and P_{i+1}, there are two knot intervals, Δ_i^i and Δ_i^{i+1}, $2 \leqslant i \leqslant n-1$. In general, $\Delta_i^i \neq \Delta_i^{i+1}$. Each of the two end data points has only one knot interval, i.e., Δ_1^2 for the pair P_1 and P_2, and Δ_{n-1}^{n-1} for the pair P_{n-1} and P_n. We average the two sequences of knot intervals, $\{\Delta_i^i\}$

and $\left\{\Delta_i^{i+1}\right\}$, into a single sequence of knot intervals, $\{\Delta_i\}$, $i = 1, \ldots, n-1$, using the following formula

$$\begin{aligned} &\Delta_1 = \Delta_1^2 \\ &\Delta_i = \alpha_i \Delta_i^i + \beta_i \Delta_i^{i+1}, \quad i = 2, \ldots, n-2 \\ &\Delta_{n-1} = \Delta_{n-1}^{n-1} \end{aligned} \tag{39}$$

where α_i and β_i are weights, satisfying $\alpha_i + \beta_i = 1$.

We now discuss the computation of α_i and β_i in Eq. (39). If all the data points are taken from the same quadratic curve, then

$$\begin{aligned} &\alpha_i = \beta_i = 0.5 \\ &\alpha_i \Delta_i^i - \beta_i \Delta_i^{i+1} = 0 \end{aligned} \tag{40}$$

If all data points are not taken from the same quadratic curve, the values of α_i and β_i will be different, so Δ_i^i and Δ_i^{i+1} should have different effects on Δ_i. Corresponding to P_i and P_{i+1}, there are two knot intervals $1 - s_i$ and s_{i+1}. If $s_i(1 - s_i) > s_{i+1}(1 - s_{i+1})$, in general, $|d_i - d_{i-1}| < |d_{i+1} - d_i|$, which means that s_i has higher precision than s_{i+1}, so Δ_i^i should have a bigger effect on Δ_i than Δ_i^{i+1}. On the other hand, if $s_i > 1 - s_i$, Δ_i^i has higher precision than Δ_{i-1}^i as in the case $d_{i-1} > d_i$, and similarly, if $1 - s_{i+1} > s_{i+1}$, Δ_i^{i+1} has higher precision than Δ_{i+1}^{i+1}. This means that α_i and β_i should be proportional to the values $s_i^2(1 - s_i)$ and $s_{i+1}(1 - s_{i+1})^2$, respectively. For convenience, we first define two knot effect factors:

$$\begin{aligned} \alpha_i^0 &= \frac{s_i^2(1 - s_i)}{s_i^2(1 - s_i) + s_{i+1}(1 - s_{i+1})^2} \\ \beta_i^0 &= \frac{s_{i+1}(1 - s_{i+1})^2}{s_i^2(1 - s_i) + s_{i+1}(1 - s_{i+1})^2} \end{aligned}$$

To determine α_i and β_i from Eq. (40), we first define the following objective function

$$G(\alpha_i^1, \beta_i^1) = (\alpha_i^1 - \alpha_i^0)^2 + (\beta_i^1 - \beta_i^0)^2 + (\alpha_i^1 \Delta_i^i - \beta_i^1 \Delta_i^{i+1})^2$$

Minimizing $G(\alpha_i^1, \beta_i^1)$ yields:

$$\begin{aligned} \alpha_i^1 &= \frac{\alpha_i^0(\Delta_i^{i+1}\Delta_i^{i+1} + 1) + \beta_i^0 \Delta_i^i \Delta_i^{i+1}}{2(\Delta_i^i)^2 + 2(\Delta_i^{i+1})^2 + 2} \\ \beta_i^1 &= \frac{\beta_i^0(\Delta_i^i \Delta_i^i + 1) + \alpha_i^0 \Delta_i^i \Delta_i^{i+1}}{2(\Delta_i^i)^2 + 2(\Delta_i^{i+1})^2 + 2}. \end{aligned} \tag{41}$$

In general, $\alpha_i^1 + \beta_i^1 \neq 1$, they can not be used to define α_i and β_i directly. The factors α_i^0 and β_i^0 will be used to define the final α_i and β_i again. Now α_i and β_i in Eq. (39) are defined by

$$\begin{aligned} \alpha_i &= \frac{\alpha_i^0 \alpha_i^1}{\alpha_i^0 \alpha_i^1 + \beta_i^0 \beta_i^1} \\ \beta_i &= \frac{\beta_i^0 \beta_i^1}{\alpha_i^0 \alpha_i^1 + \beta_i^0 \beta_i^1} \end{aligned} \tag{42}$$

For the end data points, there is only one knot interval, Δ_2^1, for the pair P_1 and P_2, and there is one knot interval, Δ_{n-1}^{n-1}, for the pair P_{n-1} and P_n. So Δ_1 and Δ_{n-1} are defined by

$$\begin{aligned} &\Delta_1 = \Delta_2^1 \\ &\Delta_{n-1} = \Delta_{n-1}^{n-1} \end{aligned} \tag{43}$$

Now, the global knot sequence $\{t_i\}$, $i = 1, \ldots, n$, is determined by

$$\begin{aligned} &t_1 = 0 \\ &t_{i+1} = t_i + \Delta_i, \quad i = 1, \ldots, n-1 \end{aligned} \tag{44}$$

5 Experiments

Various experiments are presented in this section, including comparisons between our method (New) with the explicit function method (M0) [18], the chord length method (M1), Foley's method (M2), the centripetal method (M3), the quadratic polynomial precision method (M4) [15], the rational chord length method (M5) [27], and the refined centripetal method (M6) [12]. The comparison is carried out using three types data points sampled from three sets of primitive curves. For consistency, two are taken from existing studies [14, 15]. The 8 methods are compared by computing knots for constructing interpolation curves. The method of constructing curves and computing the tangent vector at each point follow Li et al. [18]. Afterwards, we compare the interpolation precision of piecewise cubic Hermite curves constructed by these 8 methods, and consider the performance of the algorithms.

Data points of the first type are sampled from a family of ellipse arcs, $F_1(k,t) = (x_1(k,t), y_1(k,t))$, defined by

$$\begin{aligned} x_1(k,t) &= (2 + 0.5k)\cos(2\pi t) \\ y_1(k,t) &= 2\sin(2\pi t) \end{aligned} \tag{45}$$

where $k = 0, \ldots, 13$. Data points of the second type are sampled from a family of cubic Hermite curves, $F_2(k,t) = (x_2(k,t), y_2(k,t))$, $k = 1, \ldots, 14$, defined by

$$\begin{aligned} x_2(k,t) &= df_1(t) + 3g_0(t) + dg_1(t) \\ y_2(k,t) &= df_1(t) - dg_1(t) \end{aligned} \tag{46}$$

where $d = 3 + 0.5k$, and $f_0(t)$, $f_1(t)$, $g_0(t)$, $g_1(t)$ are cubic Hermite basic functions on $[0, 1]$.

$$\begin{aligned} f_0(t) &= (1-t)^2(1+2t), \quad & f_1(t) &= (1-t)^2 \\ g_0(t) &= t^2(3-2t), & g_1 &= -t^2(1-t) \end{aligned}$$

The knots computed by the new method and method M4 are exact when the data points are taken from a quadratic polynomial curve, and since $F_2(k,t)$ is a quadratic polynomial at $k=0$, the case when $k=0$ is discarded here. Plots of $F_2(k,t)=(x_2(k,t),y_2(k,t))$, $k=0,2,4,\ldots,14$, are given in Fig. 4.

Data points of the third type are taken from four basic curves, $F_l(t)=(x_l(t),y_l(t))$, $l=3,4,5,6$, defined as follows:

$$\begin{aligned} x &= t \\ y &= \sin(\pi t) \end{aligned} \tag{47}$$

$$\begin{aligned} x &= t \\ y &= \mathrm{e}^{\pi t} \end{aligned} \tag{48}$$

$$\begin{aligned} x &= t \\ y &= \sqrt{1+(\pi t)^2} \end{aligned} \tag{49}$$

$$\begin{aligned} x &= t \\ y &= \frac{1}{1+(t-0.5)^2} \end{aligned} \tag{50}$$

For making the comparison, the interval [0, 1] is divided into 20 sub-intervals to define the data points $P_i=F_j(k,t_i)$ or $F_l(t_i)$, $i=0,1,\cdots,19$, $j=1,2$, $l=3,\ldots,6$, where t_i is defined by

$$t_i=[i+\lambda\sin((20-i)i)],\quad i=0,1,\cdots,20 \tag{51}$$

where $0<\lambda\leqslant 0.25$ to ensure the data points are non-uniformly distributed [14, 15], and meet $\max\{d_{i-1},d_i\}\leqslant 3\min\{d_{i-1},d_i\}$.

As $F_2(k,t)$ and $F_l(t)$, $l=3,\ldots,6$ are not closed curves, it is therefore easy to reach the maximum error at the end points. Instead, the tangent vectors of $F_2(k,t)$ and $F_l(t)$, $l=3,\ldots,6$, at the end points $t=0$ and $t=1$ are used to construct the cubic Hermite curves. The absolute errors of $F_1(k,t)$, $F_2(k,t)$, and $F_l(t)$, $l=3,\ldots,6$, are used to evaluate the algorithms' performance, defined as below [14, 15]:

$$\begin{aligned} E_j(k,t) &= |P(s)-F_j(k,t)| \\ &= \min\{|P_i(s)-F_j(k,t)|\},\quad j=1,2 \\ E_l(t) &= |P(s)-F_l(t)| \\ &= \min\{|P_i(s)-F_l(t)|\},\quad l=3,4,5 \\ &\quad s_i\leqslant s\leqslant s_{i+1},\quad i=0,1,\ldots,19 \end{aligned} \tag{52}$$

where $P(s)$ denotes one of the cubic Hermite curves constructed by the 8 methods. $F_j(k,t)$, $j=1,2$, and $F_l(t)$, $l=3,4,5,6$, are defined by Eqs. (45)–(50), respectively. $P_i(s)$ denotes the part of $P(s)$ on the interval $[s_i,s_{i+1}]$. The distance from $P(s)$ to $F_j(k,t)$ and $F_l(t)$ is defined as $|P(s)-F_j(k,t)|$, $j=1,2$, and $|P(s)-F_l(t)|$, $l=3,4,5,6$.

Results of comparing these 8 methods for the first and second types of data points are given first. Tables 1 and 2 give the maximum values of errors $E_1(k,t)$ and $E_2(k,t)$ generated by the 8 methods, where in $E_1(k,t)$, $k=0,1,\cdots,13$, and in $E_2(k,t)$, $k=1,\cdots,14$, when $\lambda=0.15$ in Eq. (51). In these tables, minimum values of maximum errors are highlighted. They clearly demonstrate that the new method has lowest maximum error in most cases. Figure 5 gives the error curves $E_1(k,t)$ and $E_2(k,t)$ at $k=6$, $\lambda=0.15$ produced by the various methods, which gives a visual understanding of the precision of the curves constructed by these methods. The error curves for M5 is not shown because of its similarity to the ones for M3. These results indicate the higher precision of the curves constructed by the new method, trailed by the M0 and M4 methods.

Results on further experiments on the third type of data points, sampled from four basic curves defined by Eqs. (47)–(50), are provided in Table 3. It gives the maximum errors for the set of data points sampled from $F_1(t)$, when $\lambda=0.05i$, $i=1,\ldots,5$ in Eq. (51). Table 3 reveals that, for $F_1(t)$, the precision of the new method is much greater than for the other 7 methods. Similar comparison experiments for the other $F_l(t)=(x_l(t),y_l(t))$, $l=4,5,6$, show similar results. Table 3 also indicates that, when constructing curves interpolating the third type of data points, the new method has an obvious advantage in curve precision over the other seven methods. Among the rest seven methods, M0 provides better results than M1–M6.

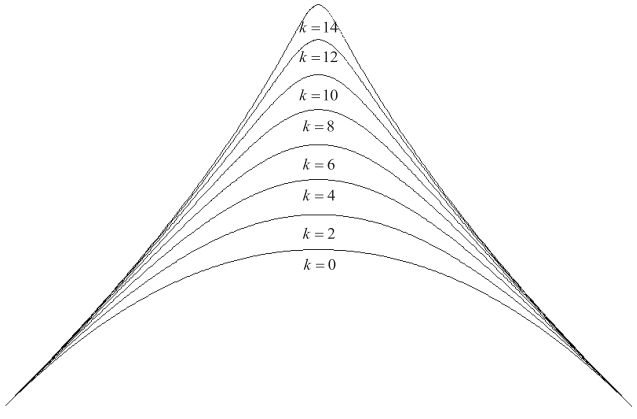

Fig. 4 Plots of $F_2(k,t)$.

Table 1 Maximum errors of $E_1(k,t)$ for $\lambda = 0.15$

$E_1(k,t)$	New	M0	M1	M2	M3	M4	M5	M6
$K=0$	1.33e-3	1.64e-3	**1.02e-3**	5.95e-3	1.03e-2	1.11e-3	1.76e-2	8.96e-3
$K=1$	1.37e-3	1.78e-3	2.18e-3	7.67e-3	1.24e-2	**1.24e-3**	2.09e-2	1.09e-2
$K=2$	**1.71e-3**	2.04e-3	3.49e-3	9.14e-3	1.40e-2	1.75e-3	2.36e-2	1.23e-2
$K=3$	**2.09e-3**	2.39e-3	5.59e-3	1.03e-2	1.51e-2	2.24e-3	2.58e-2	1.32e-2
$K=4$	**2.46e-3**	2.70e-3	7.99e-3	1.14e-2	1.57e-2	2.71e-3	2.77e-2	1.36e-2
$K=5$	**2.79e-3**	2.96e-3	1.07e-2	1.27e-2	1.58e-2	3.19e-3	2.93e-2	1.37e-2
$K=6$	**3.10e-3**	3.19e-3	1.35e-2	1.42e-2	1.60e-2	3.63e-3	3.06e-2	1.50e-2
$K=7$	**3.38e-3**	3.39e-3	1.73e-2	1.55e-2	1.76e-2	4.03e-3	3.17e-2	1.66e-2
$K=8$	3.64e-3	**3.58e-3**	2.20e-2	1.66e-2	1.91e-2	4.41e-3	3.26e-2	1.82e-2
$K=9$	3.88e-3	**3.75e-3**	2.71e-2	1.76e-2	2.04e-2	4.75e-3	3.34e-2	1.97e-2
$K=10$	4.10e-3	**3.93e-3**	3.23e-2	1.84e-2	2.17e-2	5.08e-3	3.40e-2	2.10e-2
$K=11$	4.30e-3	**4.22e-3**	3.76e-2	1.91e-2	2.28e-2	5.38e-3	3.45e-2	2.21e-2
$K=12$	**4.58e-3**	4.71e-3	4.30e-2	1.97e-2	2.38e-2	5.66e-3	3.50e-2	2.30e-2
$K=13$	**4.97e-3**	5.51e-3	4.83e-2	2.02e-2	2.46e-2	5.93e-3	3.54e-2	2.39e-2

Table 2 Maximum errors of $E_2(k,t)$ for $\lambda = 0.15$

$E_2(k,t)$	New	M0	M1	M2	M3	M4	M5	M6
$K=1$	2.23e-5	4.18e-5	7.87e-5	2.24e-4	8.09e-4	**2.15e-5**	1.26e-4	7.60e-4
$K=2$	**4.58e-5**	5.49e-5	1.01e-4	2.76e-4	8.83e-4	4.64e-5	1.57e-4	8.32e-4
$K=3$	**7.15e-5**	7.69e-5	1.24e-4	3.31e-4	9.53e-4	7.37e-5	1.78e-4	9.01e-4
$K=4$	**9.99e-5**	1.03e-4	1.63e-4	3.88e-4	1.02e-3	1.06e-4	1.90e-4	9.64e-4
$K=5$	**1.33e-4**	**1.33e-4**	2.06e-4	4.45e-4	1.07e-3	1.51e-4	2.30e-4	1.02e-3
$K=6$	**1.31e-4**	1.67e-4	2.47e-4	4.99e-4	1.12e-3	1.85e-4	2.66e-4	1.06e-3
$K=7$	**1.82e-4**	2.05e-4	2.83e-4	5.49e-4	1.14e-3	1.93e-4	2.97e-4	1.09e-3
$K=8$	**2.09e-4**	2.45e-4	3.76e-4	5.90e-4	1.15e-3	2.49e-4	3.24e-4	1.10e-3
$K=9$	3.34e-4	**2.86e-4**	4.92e-4	6.58e-4	1.12e-3	5.03e-4	3.46e-4	1.07e-3
$K=10$	**3.16e-4**	3.33e-4	6.39e-4	7.23e-4	1.04e-3	4.18e-4	3.97e-4	1.01e-3
$K=11$	**3.49e-4**	3.97e-4	9.32e-4	7.76e-4	9.93e-4	5.09e-4	4.70e-4	9.73e-4
$K=12$	**4.44e-4**	4.50e-4	1.35e-3	8.06e-4	1.03e-3	5.78e-4	5.54e-4	1.02e-3
$K=13$	**3.84e-4**	6.19e-4	1.88e-3	7.97e-4	1.04e-3	5.42e-4	6.48e-4	1.04e-3
$K=14$	**4.99e-4**	8.43e-4	2.50e-3	7.23e-4	9.85e-4	8.13e-4	7.48e-4	9.79e-4

Table 3 Maximum errors of $F_1(t)$

$F_1(t)$	New	M0	M1	M2	M3	M4	M5	M6
$\lambda = 0.05$	**4.28e-5**	1.56e-4	4.15e-4	2.87e-4	3.87e-4	5.29e-5	2.80e-4	5.46e-4
$\lambda = 0.10$	**4.51e-5**	1.64e-4	4.25e-4	4.52e-4	6.09e-4	5.68e-5	3.08e-4	1.03e-3
$\lambda = 0.15$	**4.82e-5**	1.73e-4	4.32e-4	6.19e-4	9.59e-4	6.05e-5	3.39e-4	1.58e-3
$\lambda = 0.20$	**5.75e-5**	1.82e-4	4.36e-4	8.46e-4	1.36e-3	6.38e-5	3.71e-4	2.17e-3
$\lambda = 0.25$	7.27e-5	1.90e-4	4.37e-4	1.13e-3	1.82e-3	**6.68e-5**	4.09e-4	2.83e-3

6 Conclusions

The discussion in this paper shows that computing the knots for a given set of data points can be reduced to the problem of constructing a quadratic or cubic polynomial curve. Our new method is based on the fact that the curve segment between four adjacent points can be approximated by a quadratic polynomial or a cubic polynomial. If four adjacent consecutive data points form a convex polygon, they determine a unique quadratic polynomial interpolation curve. If the four data points do not form a convex polygon, a cubic polynomial curve with one free variable is used to interpolate the four points; the variable is determined by minimizing the cubic coefficient of the curve. Doing so makes the methods for constructing quadratic and cubic polynomials consistent, in that the cubic coefficient for the constructed quadratic polynomial curve is zero. Minimizing the cubic coefficient of the cubic polynomial curve makes the cubic polynomial curve approximate the polygon composed of the four data points well, and hence makes the curve have the

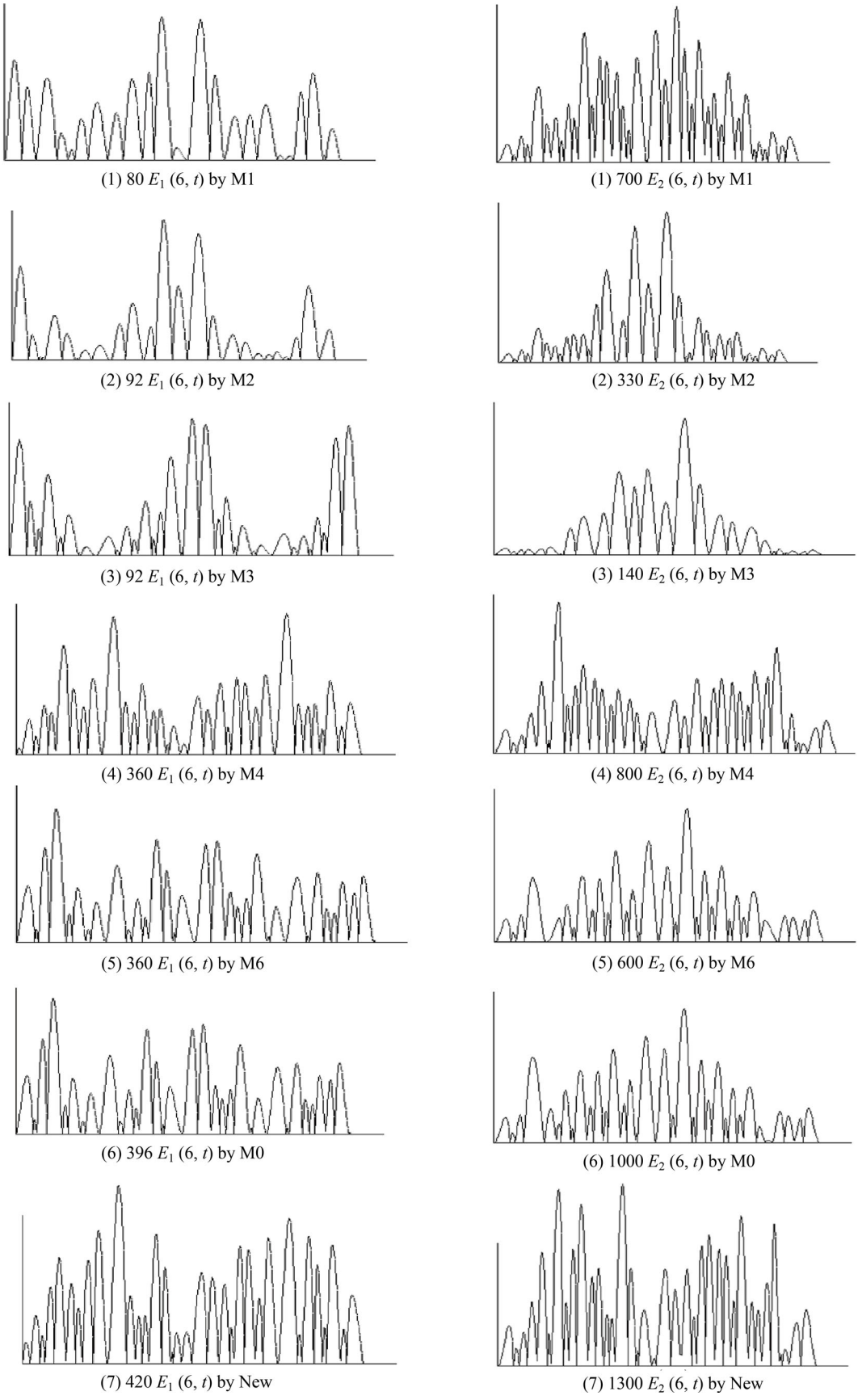

Fig. 5 Error curves for six methods.

shape suggested by the four data points. As the knots are determined by the quadratic curve or the cubic curve, they reflect the distribution of the data points. After the quadratic and cubic polynomial functions have been determined, computing the knot for each data point is an easy task. One of the advantages of the new method is that the knots have quadratic polynomial precision, while the ones proposed in Refs. [7–9, 12, 18, 27] have only linear precision. This means that from an approximation point of view, the new method, like the one in Ref. [15], is better than the other six methods. Therefore, when used for curve construction, the resulting curve has higher precision than for methods with linear precision. The second advantage of the new method is that it is affine invariant which is very important. Furthermore, our method is local, so it is easy to modify a curve interactively, consequently making the curve design process efficient and flexible. Experiments verify that approximation precision with our method is better than for the ones proposed in Refs. [7–9, 12, 15, 18, 27].

It is known that, when constructing a cubic spline interpolant, with suitable end conditions and knots, the constructed parametric cubic spline reproduces parametric cubic polynomials. Our next plan is to investigate whether there is a method of choosing knots with cubic precision. We also intend to extend the new method to data parameterization for constructing surfaces to fit scattered data points, so that for each local region, the parameters associated with the data points are computed using a local method, and the constructed surface has GC^1 continuity.

Acknowledgements

This work was supported in part by the following: National Natural Science Foundation of China under Grant Nos. 61602277 and 61772319, Natural Science Foundation of Shandong Province under Grant Nos. ZR2016FQ12 and ZR2018BF009, Key Research and Development Program of Yantai City under Grant No. 2017ZH065, CERNET Innovation Project under Grant No. NGII20161204, and Science and Technology Innovation Program for Distributed Young Talents of Shandong Province Higher Education Institutions under Grant No. 2019KJN042.

References

[1] Ahlberg, J. H.; Nilson, E.; Walsh, J. L. *The Theory of Splines and Their Applications.* Academic Press, 1967.

[2] Boor, C. D. *A Practical Guide to Splines.* Springer-Verlag New York, 1978.

[3] Brodlie, K. W. A review of methods for curve and function drawing. In: *Mathematical Methods in Computer Graphics and Design.* London: Academic Press, 33–37, 1980.

[4] Faux, I. D.; Pratt, M. J. *Computational Geometry for Design and Manufacture.* Halsted Press, 1979.

[5] Su, B. Q.; Liu, D. Y. *Computational Geometry: Curve and Surface Modeling.* Academic Press, 1989.

[6] Li, W. S.; Xu, S. H.; Zheng, J. M.; Zhao, G. Target curvature driven fairing algorithm for planar cubic B-spline curves. *Computer Aided Geometric Design* Vol. 21, No. 5, 499–513, 2004.

[7] Lü, W. Curves with chord length parameterization. *Computer Aided Geometric Design* Vol. 26, No. 3, 342–350, 2009.

[8] Farin, G. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, 1988.

[9] Lee, E. T. Y. Choosing nodes in parametric curve interpolation. *Computer-Aided Design* Vol. 21, No. 6, 363–370, 1989.

[10] Jeong, S. Y.; Choi, Y. J.; Park, P. Parametric interpolation using sampled data. *Computer-Aided Design* Vol. 38, No. 1, 39–47, 2006.

[11] Yuksel, C.; Schaefer, S.; Keyser, J. Parameterization and applications of Catmull–Rom curves. *Computer-Aided Design* Vol. 43, No. 7, 747–755, 2011.

[12] Fang, J. J.; Hung, C. L. An improved parameterization method for B-spline curve and surface interpolation. *Computer-Aided Design* Vol. 45, No. 6, 1005–1028, 2013.

[13] Lim, C. G. A universal parametrization in B-spline curve and surface interpolation. *Computer Aided Geometric Design* Vol. 16, No. 5, 407–422, 1999.

[14] Zhang, C. M.; Cheng, F. H. F.; Miura, K. T. A method for determining knots in parametric curve interpolation. *Computer Aided Geometric Design* Vol. 15, No. 4, 399–416, 1998.

[15] Zhang, C. M.; Wang, W. P.; Wang, J. Y.; Li, X. M. Local computation of curve interpolation knots with quadratic precision. *Computer-Aided Design* Vol. 45, No. 4, 853–859, 2013.

[16] Hartley, P. J.; Judd, C. J. Parametrization and shape of B-spline curves for CAD. *Computer-Aided Design* Vol. 12, No. 5, 235–238, 1980.

[17] Marin, S. P. An approach to data parametrization in parametric cubic spline interpolation problems. *Journal of Approximation Theory* Vol. 41, No. 1, 64–86, 1984.

[18] Li, X. M.; Zhang, F.; Chen, G. N.; Zhang, C. M. Formula for computing knots with minimum stress and stretching energies. *Science China Information Sciences* Vol. 61, No. 5, Article No. 052104, 2017.

[19] Lin, F. M.; Shen, L. Y.; Yuan, C. M.; Mi, Z. P. Certified space curve fitting and trajectory planning for CNC machining with cubic B-splines. *Computer-Aided Design* Vol. 106, 13–29, 2019.

[20] Yang, Z. Y.; Shen, L. Y.; Yuan, C. M.; Gao, X. S. Curve fitting and optimal interpolation for CNC machining under confined error using quadratic B-splines. *Computer-Aided Design* Vol. 66, 62–72, 2015.

[21] Floater, M. S.; Reimers, M. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design* Vol. 18, No. 2, 77–92, 2001.

[22] Gotsman, C.; Gu, X. F.; Sheffer, A. Fundamentals of spherical parameterization for 3D meshes. In: Proceedings of the ACM SIGGRAPH 2003 Papers, 358–363, 2003.

[23] Gu, X. F.; Yau, S. T. Global conformal surface parameterization. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 127–137, 2003.

[24] Xie, H.; Qin, H. A novel optimization approach to the effective computation of NURBS knots. *International Journal of Shape Modeling* Vol. 7, No. 2, 199–227, 2001.

[25] Farin, G. Rational quadratic circles are parametrized by chord length. *Computer Aided Geometric Design* Vol. 23, No. 9, 722–724, 2006.

[26] Bastl, B.; Jüttler, B.; Lávička, M.; Schicho, J.; Šír, Z. Spherical quadratic Bézier triangles with chord length parameterization and tripolar coordinates in space. *Computer Aided Geometric Design* Vol. 28, No. 2, 127–134, 2011.

[27] Bastl, B.; Jüttler, B.; Lávička, M.; Šír, Z. Curves and surfaces with rational chord length parameterization. *Computer Aided Geometric Design* Vol. 29, No. 5, 231–241, 2012.

[28] Tsuchie, S.; Okamoto, K. High-quality quadratic curve fitting for scanned data of styling design. *Computer-Aided Design* Vol. 71, 39–50, 2016.

[29] Han, X. L. A class of general quartic spline curves with shape parameters. *Computer Aided Geometric Design* Vol. 28, No. 3, 151–163, 2011.

[30] Bashir, U.; Abbas, M.; Ali, J. M. The G^2 and C^2 rational quadratic trigonometric Bézier curve with two shape parameters with applications. *Applied Mathematics and Computation* Vol. 219, No. 20, 10183–10197, 2013.

[31] Antonelli, M.; Beccari, C. V.; Casciola, G. High quality local interpolation by composite parametric surfaces. *Computer Aided Geometric Design* Vol. 46, 103–124, 2016.

Fan Zhang received his B.S. and Ph.D. degrees in computer science from Shandong University in 2009 and 2015, respectively. From 2012 to 2014, he visited the Department of Computer Science, University of Kentucky, USA, as a joint-training Ph.D. student. He is currently an associate professor with the School of Computer Science and Technology, Shandong Business and Technology University. His research interests include image processing, computer vision, computer graphics, and CAGD.

Jinjiang Li received his B.S. and M.S. degrees in computer science from Taiyuan University of Technology, China, in 2001 and 2004, respectively, and his Ph.D. degree in computer science from Shandong University, Jinan, China, in 2010. From 2004 to 2006, he was an assistant research fellow with the Institute of Computer Science and Technology, Peking University. From 2012 to 2014, he was a post-doctoral fellow with Tsinghua University, Beijing. He is currently a professor with the School of Computer Science and Technology, Shandong Technology and Business University. His research interests include image processing, computer graphics, computer vision, and machine learning.

Peiqiang Liu received his Ph.D. degree in computer software and theory from Shandong University in 2013. He is currently a professor at Shandong Technology and Business University. His research interests include algorithms and complexity theory, and computational biology.

Hui Fan received his B.S. degree in computer science from Shandong University in 1984 and his Ph.D. degree in computer science from Taiyuan University of Technology in 2007. From 1984 to 2001, he was a professor at the Computer Department of Taiyuan University of Technology. He is currently a professor at Shandong Technology and Business University. His research interests include computer aided geometric

design, computer graphics, information visualization, virtual reality, and image processing.