Interpolatory Curve Modeling with Feature Points Control

Zhonggui Chen[a], Jinxin Huang[b,c], Juan Cao[b,c,*], Yongjie Jessica Zhang[d]

[a]*Department of Computer Science, Xiamen University, Xiamen, 361000, China*
[b]*School of Mathematical Sciences, Xiamen University, Xiamen, 361005, China*
[c]*Fujian Provincial Key Laboratory of Mathematical Modeling and High-Performance Scientific Computation, Xiamen University, Xiamen, 361005, China*
[d]*Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Abstract

In curve modeling, interpolation allows users to directly control the location of curve features. While previous literatures have focused on generating interpolatory curves with properties of smoothness and locality, they usually have difficulty in controlling over geometric feature points (e.g., cusps, loops, and inflection points) that mark salient intrinsic features of curve shapes. In this paper, we propose an intuitive and efficient method for constructing planar cubic curves that are curvature continuous almost-everywhere and interpolate a sequence of input data points. Our method provides a good control over the location and type of the geometric feature points. In particular, cusps and loops only occur at specified input data points, while inflection points only occur at specified input data points and joints. We refer to such feature points controlled interpolatory curves as FPC-Curves for short. To construct FPC-Curves, we focus on piecewise cubic curves, where the occurrence of loops, cusps and inflection points is mutually exclusive. We also provide a simple yet efficient algorithm for real-time interactive design of cubic FPC-Curves. Various experimental results show the efficacy and flexibility of our new approach for curve modeling.

Keywords: Interpolation, feature points, curve modeling.

1. Introduction

Planar curve modeling has wide-range applications in geometric design, including computer graphics, animation, computer aided geometric design, and computer numerical control. There are basically three techniques for planar curve modeling: interpolation, approximation and direct manipulation of Bézier/B-spline control points. Among them, interpolation is a popular technique through a sequence of relatively sparse data points. These data points are input from designers for resulting shape control. The most important advantage of interpolation over the other two methods is its direct association between the input and the resulting curves. In the literature, people focus on different properties of interpolating curves, such as fairness, extensionality, monotone curvature and locality.

In this paper, we focus on constructing smooth 2D curves controlled by salient geometric feature points (including cusps, loops, and inflection points) through a given set of sparse data points. Each resulting curve segment will pass through a given data point, where the data point becomes a cusp, a loop, an inflection point or a regular point as pre-specified. We refer to such feature points controlled interpolatory curves as FPC-Curves for short. This design task domain is quite different from the problem arising from reverse engineering and computer numerical control, where the curves with cusps, inflection points, and loops should be avoided. The primary application envisioned for our work is more for artistic design, where geometric feature points marking salient intrinsic features are desired; see Figure 1.

We should point out that salient geometric feature points can also be created by previous interpolation methods. However, it is usually less-intuitive and labor intensive for users to control the

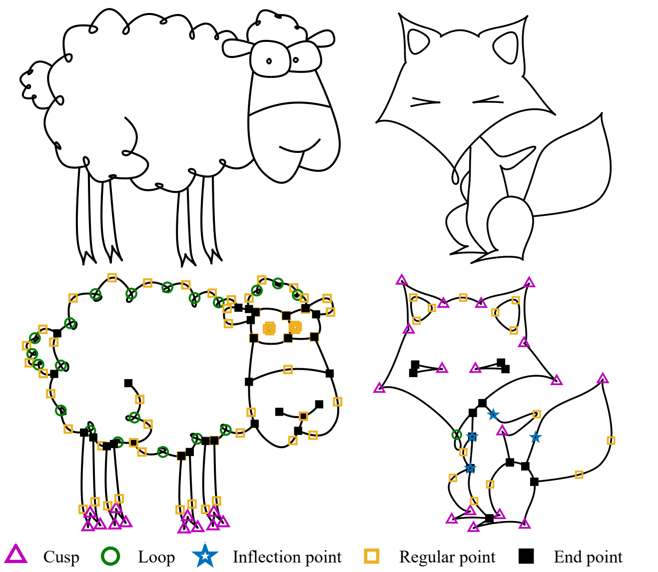

Figure 1: Our method constructs cubic FPC-Curves (top row) that are G^2 continuous almost everywhere and interpolate sequences of input geometric feature points (bottom row).

location of feature points, and it may introduce unwanted cusps or loops; see Figure 2. In this paper, we propose to create interpolatory FPC-Curves using cubic Bézier segments, where each segment contains at most one interior feature point at the user-specified position. Our method creates a concatenation of cubic Bézier curves joining G^2 continuously almost everywhere except at joints with sign change curvature, where curves are G^1 continuous. Differing from most applications in CAD, where G^2 continuity is usually desired, the G^1 continuity of our FPC-Curves is

*Corresponding author
Email address: juancao@xmu.edu.cn (Juan Cao)

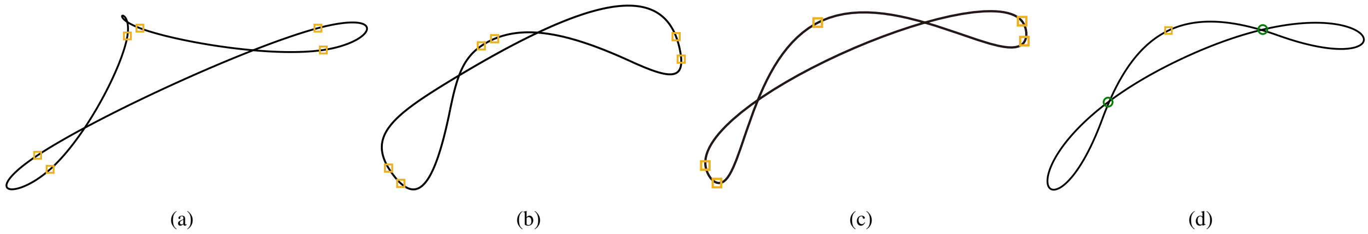

Figure 2: Comparison of our FPC-Curves with other interpolatory curves in modeling loops. (a) C^2 interpolatory cubic B-spline [1], which creates a small loop away from the input data points; (b) cubic Bézier curves interpolating 6 regular points generated by our method; (c) interpolatory quadratic Bézier curves [2], where at least 5 input data points are needed to create two loops, and it is hard to control the location of loops; (d) our FPC-Curve, where only 3 control points are needed to model two loops that occur at two input data points (green circles). Orange squares and green circles denote the input data points.

good enough for artistic design. In particular, the specific contributions of this paper are as follows:

1. We propose an intuitive and efficient method for constructing cubic FPC-Curves that are G^2 continuous almost everywhere and interpolate a sequence of control points. This method provides a good control over the location and type of the geometric feature points. In particular, cusps and loops only occur at control points, while inflection points only occur at control points and joints.

2. We tailor an iterative optimization algorithm to achieve the real-time interactive design. To create interpolatory curves with satisfying properties, we focus on cubic Bézier segments, which are the lowest degree polynomial curves that take on loops, cusps, or inflection points within a single segment. We study the geometric conditions for cubic curves (having loops, cusps, or inflection points) and G^2 continuity, based on which we provide a simple yet efficient optimization framework that creates desired FPC-Curves.

The remainder of this paper is organized as follows. Section 2 reviews the related work. The geometric constraints for a parametric cubic curve taking on a loop, cusp, or inflection points, and G^2 continuous constraints are studied in Section 3. Section 4 presents the algorithm and implementation for our interpolatory FPC-Curve construction. Results and discussion are presented in Section 5. Finally, in Section 6 conclusions and future work are proposed.

2. Related work

A large number of spline interpolation methods have been developed for constructing planar curves from a sequence of input data points. A comprehensive survey is given by [3]; we only focus on the most relevant previous work here.

The properties desired for planar interpolating splines may vary in different applications. Fairness or smoothness is generally the most important property in almost all applications. One way to generate smooth splines is to optimize metrics that correlate reasonable wellness with the perception of fairness, e.g., the total bending energy stored in the spline [4]. This approach may compromise with other important properties, such as locality, roundness and stability [5]. Another way to create smooth splines is to impose constraints on the degree of continuity, a concrete property that relates to the fairness. The continuity can be evaluated either parametrically or geometrically [6], corresponding to C^k and G^k, respectively. Although continuity is not the same as fairness, there is a widespread agreement that the higher degree of continuity a curve possesses, the fairer the curve looks. In practice, C^2 or G^2 (curvature continuity) is demanded for a fair spline. Piecewise quadratic splines [7], cubic splines [1, 8] or even subdivision curves [9] are usually exploited in the interpolatory curve construction to achieve C^2 or G^2 continuity. However, these curves either introduce unexpected cusps/loops away from control points or have difficulty in controlling the location and type of feature points.

Curvature is also an objective mathematical property that is intimately tied to fairness. Some literature focus on generating curves with minimum curvature variation, e.g., minimum variation curves (MVCs). Most MVCs are constructed by approximation, such as using quantic polynomials [10]. Curvature plot consisting of relatively few monotone pieces is considered to be another fairness metric [11]. As minima and maxima of curvature are also argued to be salient features, it is suggested that all curvature extrema should coincide with the given data points in the interpolation [1]. A fair amount of literature addresses the problem of constraining curves to have monotone curvatures, e.g., Logarithmic spirals [12], Euler spirals (also known as Clothoids or Cornu spirals) [13] and class A curves [14, 15]. These curves can be adopted in data interpolation such that curvature of in-between spline segments varies monotonously. As monotone curvature segments are not as flexible as the usual Bézier/B-spline curves, it is not easy to manipulate them. Piecewise quadratic curves are used in interpolating control points with local maxima of curvature only at the input data points [2]. Since these curves are free of cusps, loops and inflection points in the interior of each segment, modeling feature points at control points is less intuitive and labor-intensive in the sense that more data points are needed; see Figure 2(c-d).

In interactive artistic design of fair curves, the number, location, and type of feature points that mark the salient shape features are required by the designer. Hence, it is reasonable to expect the designer to place these points first, and then the splines accommodate the designer's requirements by preserving these feature points. The lack of easy control of feature points inspires us to provide a method for constructing aesthetic curves with a good control over feature points. To achieve this goal, we adopt cubic Bézier curves, which are of the lowest degree to have interior cusps, loops, or inflection points. These curves have limited diversity of their characterizations. That is to say, a non-degenerate parametric cubic curve can have a cusp, a loop, or zero to two inflection points, and their occurrence is mutually exclusive [16, 17]. There are several methods for determining whether a parametric cubic curve has any loops, cusps, or inflection points. For instance, an algorithm based on algebraic properties of polynomial coefficients [16] was developed, where some geometric tests using B-spline control polygons were also included. In [17] and [18], geometric methods were presented for determining whether a cubic curve has any loops, cusps, or inflection points. By considering the position of one control point, the method in [18] is simpler and more instructive, which analyzes the lengths of tangent vectors at the ends. Our analysis of geometric constraints is based on [18].

3. Geometric constraints

For interpolatory curves, here we clarify the required geometric properties and discuss their corresponding geometric conditions. Given a sequence of data points $\boldsymbol{v}_i \in \mathbb{R}^2$, $i = 1, \cdots, N$, with specified types (including cusps, loops, inflection points, and regular points), we construct a curvature continuous (G^2 continuous) FPC-Curve that interpolates $\boldsymbol{v}_i$ and satisfies the following two requirements: (1) each curve segment interpolates one point $\boldsymbol{v}_i$, and the interpolated point $\boldsymbol{v}_i$ becomes a point on the curve with a specified type; and (2) cusps and local loops occur only at the given points and inflection points occur only at the given points or at joints. This is based on the assumption that cusps, loops, inflection points are salient feature points of a curve that should be directly controlled by the user. To make our interpolation more versatile in modeling shapes, we allow the curve segment to interpolate a regular point without introducing any feature points in that segment. We focus on cubic Bézier curves and assume to construct a closed curve in the rest of this section. Construction of open curves will be discussed in Section 4. We first recall how to determine if a parametric cubic curve has any feature points. In this paper, we use the term *feature points* to refer to the loops, cusps, or inflection points. Then we study the geometric conditions for a single Bézier segment to interpolate at a feature point or a regular point. Finally, we study the conditions for piecing together these curve segments to be curvature continuous.

3.1. Geometric characterization of parametric cubic curves

A number of previous literatures have investigated the conditions for parametric curves to have any feature points. In this section, we recall the results for parametric cubic curves. To ease the later discussion, we use the general representation of a planar parametric cubic curve $\boldsymbol{Q}(t)$:

$$\boldsymbol{Q}(t) = (x(t), y(t)) = \sum_{j=0}^{3} \boldsymbol{P}_j t^j, \tag{1}$$

where $\boldsymbol{P}_j \in \mathbb{R}^2$, $x(t)$ and $y(t)$ are cubic polynomials with derivatives $\boldsymbol{Q}'(t) = (x'(t), y'(t))$ and $\boldsymbol{Q}''(t) = (x''(t), y''(t))$. The signed curvature of the curve (1) is given by

$$\kappa(t) = \frac{x'(t)y''(t) - x''(t)y'(t)}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}}.$$

The numerator is actually a quadratic function, which can be denoted by

$$x'(t)y''(t) - x''(t)y'(t) = 2F(t) = 2(At^2 + Bt + C),$$

where $A = 3\det(\boldsymbol{P}_2, \boldsymbol{P}_3)$, $B = 3\det(\boldsymbol{P}_1, \boldsymbol{P}_3)$ and $C = \det(\boldsymbol{P}_1, \boldsymbol{P}_2)$. Note that, $F(t)$ is proportional to the signed curvature of the curve.

It has been shown that a non-degenerate (i.e., the control points are not collinear or coincident) parametric cubic curve can only have a cusp, a loop, or up to two inflection points, and the presence of loops, cusps or inflection points is mutually exclusive. In particular, whether a cubic curve has any cusps, loops, or inflection points can be entirely determined from the discriminant $\Delta = B^2 - 4AC$ of $F(t)$ as follows [16, 17]. If $A = 0$, there is exactly one inflection point. Otherwise,

- if $\Delta > 0$, there are exactly two inflection points;
- if $\Delta < 0$, there is a loop;
- if $\Delta = 0$, there is a cusp.

3.2. Interpolation of cubic Bézier curves at feature points

Like many interpolation methods, we adopt planar cubic Bézier curves. A cubic Bézier is defined as a linear combination of four control points and basis functions:

$$\boldsymbol{Q}(t) = \sum_{i=0}^{3} d_i(t)\boldsymbol{Q}_i, \quad t \in [0, 1], \tag{2}$$

where $\boldsymbol{Q}_i$ are the control points and $d_i = \frac{3!}{i!(3-i)!} t^i (1-t)^{3-i}$ are the basis functions. We can always convert a cubic Bézier curve to the general representation (1), where the control points can be computed as

$$\begin{bmatrix} \boldsymbol{P}_0 \\ \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \\ \boldsymbol{P}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_0 \\ \boldsymbol{Q}_1 \\ \boldsymbol{Q}_2 \\ \boldsymbol{Q}_3 \end{bmatrix}.$$

In other words, the cubic Bézier curve is a segment (corresponding to the parametric interval $[0, 1]$) of the general cubic curve (1) with control points defined above. Note that, $\boldsymbol{P}_0$ in the general form coincides with the control point $\boldsymbol{Q}_0$ in the Bézier form. The cubic Bézier curve (1) interpolates $\boldsymbol{v} = (x, y)$ at a specified parameter $t^* \in (0, 1)$ satisfying

$$\boldsymbol{Q}(t^*) = \boldsymbol{P}_3 {t^*}^3 + \boldsymbol{P}_2 {t^*}^2 + \boldsymbol{P}_1 t^* + \boldsymbol{P}_0 = \boldsymbol{v}. \tag{3}$$

In the following, we will discuss the additional geometric conditions for a cubic Bézier curve (2) interpolating $\boldsymbol{v}$ such that $\boldsymbol{v}$ becomes either a cusp, a loop, an inflection point, or a regular point without introducing any other feature points on the curve segment. Note that in Section 3.1, the conditions for a cubic curve presenting feature points are determined without restricting the range of parameter t. Here, we adapt the results to the Bézier representation by restricting t to $[0, 1]$. Let us first assume $A \neq 0$. We defer discussing the case of $A = 0$ until Section 4.

Cusp. A general cubic curve (1) has a cusp if and only if

$$\Delta = B^2 - 4AC = 0. \tag{4}$$

If we assign a value $t^* \in (0, 1)$ to the parameter of a cusp, then t^* should also be the root of $F(t)$. We have

$$t^* = \frac{-B}{2A}, \tag{5}$$

where A, B satisfy Eq. (4). Conditions (4-5) lead to (see Appendix A.1 for a more detailed discussion):

$$3\boldsymbol{P}_3 {t^*}^2 + 2\boldsymbol{P}_2 t^* + \boldsymbol{P}_1 = 0. \tag{6}$$

Loop. Instead of enforcing the condition $\Delta < 0$, we could enforce a more direct and intuitive condition on the cubic Bézier curve such that it takes a loop. A cubic Bézier curve with a loop means that there are two coincident points on the curve with different parameters, for example $t_1^*, t_2^* \in (0, 1)$ with $t_1^* \neq t_2^*$. Let the cubic curve interpolate the given data $\boldsymbol{v}$ at the loop, then we have

$$\boldsymbol{P}_3 t_1^{*3} + \boldsymbol{P}_2 t_1^{*2} + \boldsymbol{P}_1 t_1^* + \boldsymbol{P}_0 = \boldsymbol{v} \tag{7}$$

and

$$\boldsymbol{P}_3 t_2^{*3} + \boldsymbol{P}_2 t_2^{*2} + \boldsymbol{P}_1 t_2^* + \boldsymbol{P}_0 = \boldsymbol{v}. \tag{8}$$

Inflection point. If $\Delta > 0$, the parabola described by $F(t)$ has two roots symmetric around $t = -\frac{B}{2A}$, both of which correspond to the parameters of the inflection points. We specify one root as $t^* \in (0, 1)$,

$$F(t^*) = 0, \tag{9}$$

and introduce a parameter h such that

$$t^* + h = -\frac{B}{2A}, \tag{10}$$

where h is restricted to $(-\infty, -\frac{t^*}{2}) \cup (\frac{1-t^*}{2}, +\infty)$. Then the second root of $F(t)$ can be guaranteed to be outside the parametric domain $[0, 1]$. Intuitively, $|h|$ is half of the distance between the pair of roots, which can be used to adjust the shapes of the parabola and the final curve segment. Conditions (9-10) lead to a linear equation (see Appendix A.2 for a more detailed discussion):

$$3(t^{*2} + 2ht^*)\boldsymbol{P}_3 + 2(t^* + h)\boldsymbol{P}_2 + \boldsymbol{P}_1 = 0. \tag{11}$$

Regular point. A general cubic parametric curve (1) has either a cusp, a loop or up to two inflection points. Note that, a cubic Bézier curve (2) is part of a general cubic curve (1) with the parametric domain restricted to $[0, 1]$. Hence, a cubic Bézier curve can have zero, one, or two inflection points, depending on whether the two distinct roots of $F(t)$ fall within the interval $[0, 1]$. We obtain a regular cubic Bézier curve by enforcing the general curve to have two inflection points, both of which lie outside of $[0, 1]$. To achieve this goal, we enforce that the quadratic function $F(t)$ has one root at -1, i.e.,

$$F(-1) = A - B + C = 0, \tag{12}$$

and the symmetry axis $t = -B/2A$ of the corresponding parabola is in-between $t = 0$ and $t = 1$. We also enforce that the curve interpolates the point $\boldsymbol{v}$ at the parameter corresponding to the symmetry axis, i.e.,

$$t^* = -\frac{B}{2A}, \tag{13}$$

where A and B satisfy Eq. (12). Conditions (12-13) lead to the following linear equation (see Appendix A.3 for a more detailed discussion):

$$3(1 + 2t^*)\boldsymbol{P}_3 - 2t^*\boldsymbol{P}_2 - \boldsymbol{P}_1 = 0. \tag{14}$$

3.3. Geometric conditions for curvature continuity

We have discussed a single cubic Bézier segment for interpolating a point where the interpolated point becomes a cusp, a loop, an inflection point, or a regular point of the curve. In this section, we aim to piece these segments together to form a curvature continuous curve.

Assume there are N connected curve segments referred in the interpolation. Hereinafter, if a symbol carries two subscripts separated by a comma, then the first subscript represents the index of a curve segment, and the second subscript indicates the index of control points in that curve segment. For instance, we denote

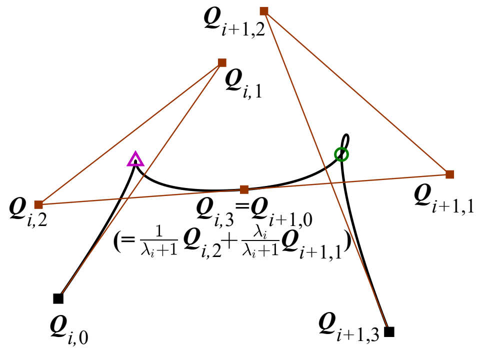

Figure 3: Conditions for G^1 continuity.

N cubic Bézier curves by $\boldsymbol{Q}_i(t)$ $(i = 1, ..., N)$ with control points $\{\boldsymbol{Q}_{i,0}, ..., \boldsymbol{Q}_{i,3}\}$, where $\boldsymbol{Q}_{i,j}$ means the j^{th} control point of the i^{th} curve segment. Moreover, the first subscript i is taken modulo N, if $i > N$.

Two sequential curve segments $\boldsymbol{Q}_i(t)$ and $\boldsymbol{Q}_{i+1}(t)$ meet with G^2 continuity if and only if they have a common end point, i.e., $\boldsymbol{Q}_{i,3} = \boldsymbol{Q}_{i+1,0} = \boldsymbol{P}_{i+1,0}$ (G^0 continuity, where $\boldsymbol{P}_{i+1,0}$ is the joint), a common unit tangent vector (G^1 continuity), i.e., $\boldsymbol{Q}_{i,3}(= \boldsymbol{Q}_{i+1,0})$, $\boldsymbol{Q}_{i,2}$ and $\boldsymbol{Q}_{i+1,1}$ are collinear, and coinciding curvatures, i.e., $\kappa_i(1) = \kappa_{i+1}(0)$ [1]. The conditions for G^1 continuity can be equivalently written as

$$\boldsymbol{Q}_{i,3} = \boldsymbol{Q}_{i+1,0} = \frac{1}{\lambda_i + 1}\boldsymbol{Q}_{i,2} + \frac{\lambda_i}{\lambda_i + 1}\boldsymbol{Q}_{i+1,1},$$

where $\lambda_i = \frac{\|\boldsymbol{Q}_{i,2}\boldsymbol{Q}_{i,3}\|}{\|\boldsymbol{Q}_{i+1,0}\boldsymbol{Q}_{i+1,1}\|}$ is the ratio between the lengths of control legs adjacent to the joint $\boldsymbol{Q}_{i,3}(= \boldsymbol{Q}_{i+1,0})$; see Figure 3.

Let the general representation of $\boldsymbol{Q}_i(t)$ to be $\boldsymbol{P}_i(t) = \sum_{j=0}^{3} \boldsymbol{P}_{i,j} t^j$, then the conditions for G^2 continuity can be described in terms $\boldsymbol{P}_{i,j}$ and λ_i. More precisely, G^0 continuity requires

$$\boldsymbol{P}_{i,0} + \boldsymbol{P}_{i,1} + \boldsymbol{P}_{i,2} + \boldsymbol{P}_{i,3} = \boldsymbol{P}_{i+1,0}, \tag{15}$$

and G^1 continuity requires

$$3\boldsymbol{P}_{i,0} + 2\boldsymbol{P}_{i,1} + \boldsymbol{P}_{i,2} - 3\boldsymbol{P}_{i+1,0} = -\boldsymbol{P}_{i+1,1}\lambda_i. \tag{16}$$

Substituting conditions (15-16) into $\kappa_i(1) = \kappa_{i+1}(0)$ yields G^2 continuity condition:

$$\lambda_i = \sqrt{\frac{\left|(\boldsymbol{P}_{i,1} + 2\boldsymbol{P}_{i,2} + 3\boldsymbol{P}_{i,3}) \times (\boldsymbol{P}_{i,2} + 3\boldsymbol{P}_{i,3})\right|}{\left|\boldsymbol{P}_{i+1,2} \times (3\boldsymbol{P}_{i,0} + 2\boldsymbol{P}_{i,1} + \boldsymbol{P}_{i,2} - 3\boldsymbol{P}_{i+1,0})\right|}}. \tag{17}$$

4. Optimization

While we have discussed geometric conditions for constructing a single cubic Bézier segment that interpolates at a feature point and piecing them together with G^2 continuity, we aim to provide an optimization method that generates curves satisfying all these conditions. Let us first recall from Section 3 that the conditions for each curve segment to have desired properties are: interpolation condition (3); conditions for controlling over the type of the interpolated points, i.e., (6) for a cusp, (8) for a loop (in this case, (3) is replaced by (7)), (11) for an inflection point, and (14) for a regular point; and conditions for G^0 to G^2 continuity (15-17). All these conditions form a non-linear system with respect to the control points $\boldsymbol{P}_{i,j}$, which are difficult to solve directly. We can observe that the first four conditions are linear equations with respect to unknown control points, if the parameter t_i^*, h_i and λ_i are fixed. Instead of directly solving the nonlinear system, we therefore compute control points according to the first four conditions to achieve G^1 continuity, and then update the ratio λ_i according to condition (17) to pursue curvature continuity. These two steps are alternatively applied until a terminate condition is satisfied.

Let us consider a cubic Bézier segment $\boldsymbol{Q}_i(t)$ that is G^1 connected to the subsequent segment $\boldsymbol{Q}_{i+1}(t)$, and they interpolate at cusps. Therefore, both segments should satisfy conditions (3), (6), and (15-17). For the i^{th} segment, we assume the parameters t_i^* and λ_i are constant, then the first four conditions are linear equations with respect to the five unknowns $\boldsymbol{P}_{i,0}, \boldsymbol{P}_{i,1}, \boldsymbol{P}_{i,2}, \boldsymbol{P}_{i,3}$, and $\boldsymbol{P}_{i+1,0}$. If we solve the first three equations (i.e., Eqs. (3), (6) and (15)) for three unknowns $\boldsymbol{P}_{i,1}, \boldsymbol{P}_{i,2}$, and $\boldsymbol{P}_{i,3}$, we can represent them as linear combinations of the joints $\boldsymbol{P}_{i,0}$ and $\boldsymbol{P}_{i+1,0}$:

$$\begin{cases} \boldsymbol{P}_{i,1} = \dfrac{t_i^{*2}}{(t_i^* - 1)^2}\boldsymbol{P}_{i+1,0} - \dfrac{t_i^* + 2}{t_i^*}\boldsymbol{P}_{i,0} - \dfrac{3t_i^* - 2}{t_i^*(t_i^* - 1)^2}\boldsymbol{v}_i, \\ \boldsymbol{P}_{i,2} = -\dfrac{2t_i^*}{(t_i^* - 1)^2}\boldsymbol{P}_{i+1,0} + \dfrac{2t_i^* + 1}{t_i^{*2}}\boldsymbol{P}_{i,0} + \dfrac{3t_i^{*2} - 1}{t_i^{*2}(t_i^* - 1)^2}\boldsymbol{v}_i, \\ \boldsymbol{P}_{i,3} = \dfrac{1}{(t_i^* - 1)^2}\boldsymbol{P}_{i+1,0} - \dfrac{1}{t_i^{*2}}\boldsymbol{P}_{i,0} - \dfrac{2t_i^* - 1}{t_i^{*2}(t_i^* - 1)^2}\boldsymbol{v}_i. \end{cases} \tag{18}$$

The middle control points of the $(i+1)^{th}$ segment can be determined in a similar fashion. In particular,

$$\boldsymbol{P}_{i+1,1} = \frac{t_{i+1}^{*2}}{(t_{i+1}^* - 1)^2}\boldsymbol{P}_{i+2,0} - \frac{t_{i+1}^* + 2}{t_{i+1}^*}\boldsymbol{P}_{i+1,0} - \frac{3t_{i+1}^* - 2}{t_{i+1}^*(t_{i+1}^* - 1)^2}\boldsymbol{v}_{i+1}.$$

Simply substituting these solutions into Eq. (16), we obtain an equation only related to the joints $\boldsymbol{P}_{i,0}$, $\boldsymbol{P}_{i+1,0}$, and $\boldsymbol{P}_{i+2,0}$:

$$\frac{(t_i^* - 1)^2}{t_i^{*2}}\boldsymbol{P}_{i,0} - \left(\frac{t_i^* - 3}{t_i^* - 1} + \lambda_i \frac{t_{i+1}^* + 2}{t_{i+1}^*}\right)\boldsymbol{P}_{i+1,0} + \lambda_i \frac{t_{i+1}^{*2}}{(t_{i+1}^* - 1)^2}\boldsymbol{P}_{i+2,0}$$
$$= \frac{3t_i^* - 1}{t_i^{*2}(t_i^* - 1)}\boldsymbol{v}_i + \lambda_i \frac{3t_{i+1}^* - 2}{t_{i+1}^*(t_{i+1}^* - 1)^2}\boldsymbol{v}_{i+1}.$$

Conditions for two G^1 connected segments where each segment interpolates at a feature point or a regular point can be derived in the same fashion. In other words, each G^1 connected interpolating curve segment leads to a linear equation only with respect to three joints of the form:

$$E_{i,1}\boldsymbol{P}_{i,0} + E_{i,2}\boldsymbol{P}_{i+1,0} + E_{i,3}\boldsymbol{P}_{i+2,0} = \boldsymbol{C}_i, \quad i = 1, ..., N, \tag{19}$$

where $E_{i,1}, E_{i,2}, E_{i,3}$, and $\boldsymbol{C}_i$ are determined by parameters t_i^*, λ_i, h_i, $\boldsymbol{v}_i$, and the type of $\boldsymbol{v}_i$ (see Appendix B). These equations form a tridiagonal system, leading to a solution for the joints. Although we cannot theoretically prove the uniqueness of the solution here, we have never found a singular coefficient matrix of the linear system (19) in all our experiments. Other control points which can be represented as linear combinations of joints can also be determined. Substituting the solution into the curvature continuity condition (17) leads to an estimation for λ_i, and an update for the linear system. We then repeat these control point solving and λ_i estimation processes until convergence. The pseudo code of our FPC-Curve optimization method is shown in Algorithm 1.

Note that, to achieve G^2 continuity at a joint with opposite curvature vectors, the curvature at that joint must be zero. Theoretically, if a cubic parametric curve has a cusp or a loop, it cannot have any inflection points, i.e., points with vanished curvature. A cubic parametric curve can only have up to two inflection points. Hence, it is impossible to achieve G^2 continuity at a joint with sign change curvature in the following two cases: (1) one of the neighboring interpolated points is either a cusp or a loop; and (2) there are three consecutive interpolated points specified to be inflection points while both joints between them have sign change curvature, as a result G^2 continuity can only be achieved at one of the joints. For simplicity, we require that the absolute values of the curvature are the same if they have opposite signs. That is to say, G^1 continuity is achieved at joints with sign change curvature.

Algorithm 1 Optimization for modeling FPC-Curves

Input: a sequence of points $\{\boldsymbol{v}_i | i = 1, ..., N\}$ with specified types (cusps, loops, inflection points, regular points, or end points) and associated parameters h_i for inflection points

Output: almost everywhere G^2 continuous pieced cubic Bézier curves that interpolate the points $\{\boldsymbol{v}_i | i = 1, ..., N\}$

1: $\lambda_i \leftarrow 1, i = 1, ..., N$
2: Compute the parameter t_i^* for each point $\boldsymbol{v}_i$ using Eqs. (20-21)
3: Solve the linear system (19) for $\{\boldsymbol{P}_{i,0} | i = 1, ..., N\}$
4: Compute the other control points $\{\boldsymbol{P}_{i,j} | i = 1, ..., N; j = 1, ..., 3\}$, which are linear combinations of $\boldsymbol{P}_{i,0}$ and $\boldsymbol{v}_i$ (a special case is shown in Eq. (18))
5: Update $\{\lambda_i | i = 1, ..., N\}$ using Eq. (17)
6: **if** $\max_i |(|\kappa_i(1)| - |\kappa_{i+1}(0)|)| > 10^{-10}$ **then**
7:     Go to Step 3
8: **else**
9:     Terminate
10: **end if**

Figure 4 shows the initial, intermediate and final states of the curve evolution process. To intuitively verify the continuity of the pieced curve, we plot the curvature vectors (toward the centers of curvature and with lengths proportional to the radii of curvature) along the curve. Our algorithm usually takes a dozen or dozens of iterations to meet the terminating condition, $\max_i |(|\kappa_i(1)| - |\kappa_{i+1}(0)|)| \leq 10^{-10}$. It converges fast, as can be observed in Figure 4 that the result in the 15^{th} iteration ($\max_i |(|\kappa_i(1)| - |\kappa_{i+1}(0)|)| \approx 10^{-6}$) is very close to the final result; and $\max_i |(|\kappa_i(1)| - |\kappa_{i+1}(0)|)|$ converges to machine precision after 50 iterations. We can also observe that the constructed interpolating curve has continuously varying curvature vectors, except at the specified cusps (which are theoretically C^2 continuous indeed) and possibly at joints where the sign of curvature changes.

Parameter settings. As discussed previously, the coefficient matrix of the linear system (19) is determined by the given points $\boldsymbol{v}_i$ (with specified types), parameters t_i^* (or $t_{1,i}^*$ and $t_{2,i}^*$ for a loop), h_i (for inflection points), and λ_i. Different parameter choices lead to interpolatory curves with different shapes. In the following we will discuss appropriate parameter selection for practical implementation. First, each given data point $\boldsymbol{v}_i$ is associated with a parameter t_i^*, computed as

$$t_i^* = \frac{\|\boldsymbol{v}_{i-1}\boldsymbol{v}_i\|}{\|\boldsymbol{v}_{i-1}\boldsymbol{v}_i\| + \|\boldsymbol{v}_i\boldsymbol{v}_{i+1}\|} \in (0, 1). \tag{20}$$

If $\boldsymbol{v}_i$ is specified to be a loop, then two parameters for this loop are

$$t_{i,1}^* = t_i^* - \alpha_i \quad \text{and} \quad t_{i,2}^* = t_i^* + \beta_i, \tag{21}$$

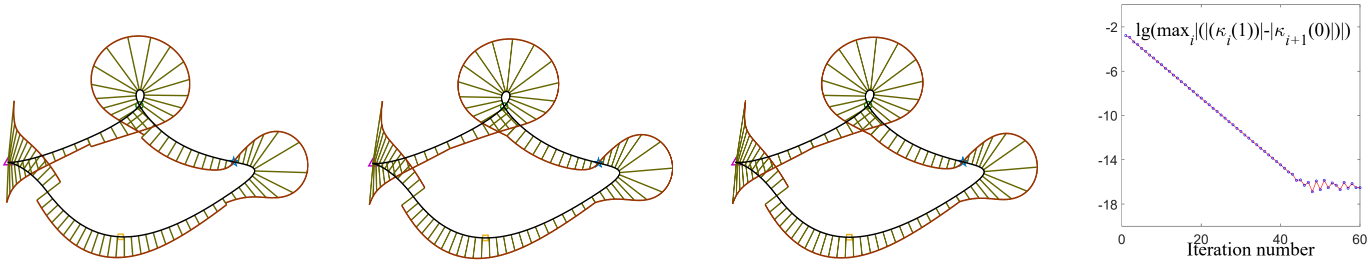

Figure 4: Curvature vectors (toward the centers of curvature and with lengths proportional to the radii of curvature) during our optimization evolving process. From left to right: our initial solution, intermediate result after 15 iterations, final result after 30 iterations, and the plot of maximum curvature error at joints vs. iteration number.

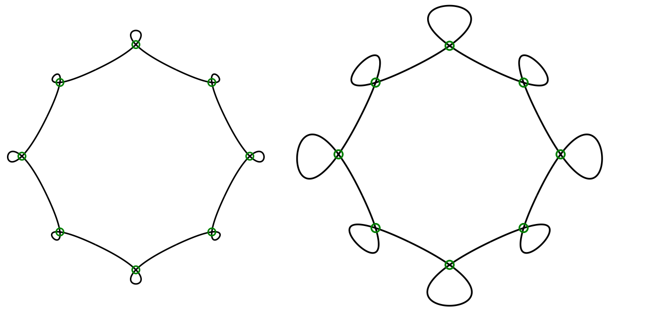

Figure 5: Size control of loops. All the α_i and β_i are set to 0.3 in the left figure, and 0.4 in the right figure.

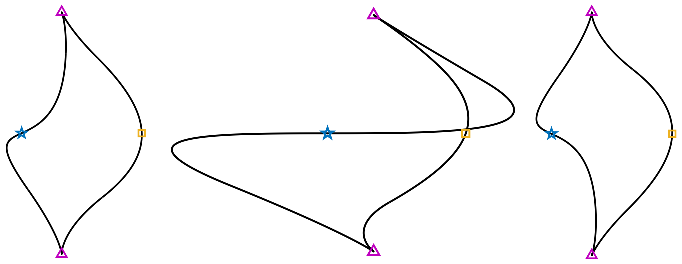

Figure 6: The sign and value of h determine the concavity of the curve and how much the curve is bending at both sides of the inflection point (marked with a blue star). Magenta triangles and orange squares denote cusps and regular data points, respectively. From left to right, h is set to be 0.5, 3, and -0.5, respectively.

where $\alpha_i \in (0, t_i^*)$ and $\beta_i \in (0, 1 - t_i^*)$ can be used to control the size of the loop. Intuitively, the size of a loop increases with the increase of $\alpha_i + \beta_i$; see Figure 5. We set $\alpha_i = \beta_i = \frac{1}{2}\min\{t_i^*, 1 - t_i^*\}$ by default. If $\boldsymbol{v}_i$ is specified to be an inflection point, the parameter h_i is chosen in the range $(-\infty, -\frac{t_i^*}{2}) \cup (\frac{1-t_i^*}{2}, +\infty)$. The sign and absolute value of h_i describe the concavity of the curve, and how much the curve is bending at both sides of the inflection points; see Figure 6. If the value of h_i is large, the curve will stretch greatly around the inflection point. We set h_i to be 0.5 by default in our implementation. Users can adjust the value of h_i to obtain a desired shape of the curve.

Open curves. In the previous discussion, we assumed that the constructed curve is closed. For a curve with end-points $\boldsymbol{v}_1$ and $\boldsymbol{v}_N$, there are only $(N-2)$ cubic Bézier segments and $(N-3)$ joints; see Figure 7. For each joint, we get a linear equation of $\boldsymbol{P}_{i,0}$, $\boldsymbol{P}_{i+1,0}$, and $\boldsymbol{P}_{i+2,0}$ as Eq. (19):

$$E_{i,1}\boldsymbol{P}_{i,0} + E_{i,2}\boldsymbol{P}_{i+1,0} + E_{i,3}\boldsymbol{P}_{i+2,0} = \boldsymbol{C}_i, \quad i = 1, ..., N-3.$$

$\boldsymbol{P}_{1,0}$ and $\boldsymbol{P}_{N-1,0}$ are specified by the end conditions:

$$\boldsymbol{P}_{1,0} = \boldsymbol{v}_1, \quad \text{and} \quad \boldsymbol{P}_{N-1,0} = \boldsymbol{v}_N.$$

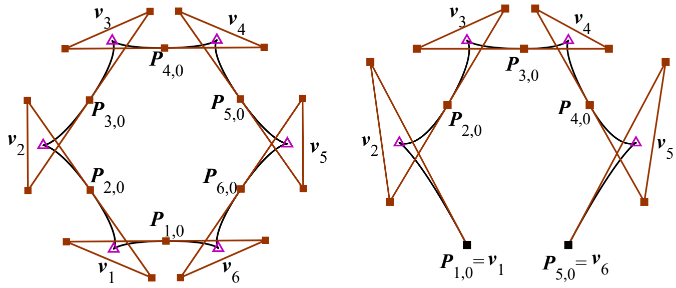

Figure 7: Left: a close curve with 6 cubic Bézier segments and 6 joints. Right: an open curve with 4 cubic cubic Bézier segments and 3 joints. They both have 6 interpolated points.

Thus we obtain a system consisting of $(N-3)$ linear equations, which is a simple modification of the system (19). Then, the unknown joints $\boldsymbol{P}_{2,0}, ..., \boldsymbol{P}_{N-2,0}$ and λ_i can be iteratively updated in the same fashion. Figures 1 and 8 show examples of curves with end-points.

Degenerate cases. Another assumption in previous discussion is that A, which is determined by control points, does not vanish. Actually when $A = 0$, the corresponding segment degenerates into a straight line segment. For instance, if the segment interpolates at a cusp, then we use the equivalent form of condition (5), $2At^* = -B$, in the implementation. Hence, $A = 0$ leads to $B = 0$ and $C = 0$, which means that all control points are collinear. The interpolation of other types of points are similar.

5. Experimental results

This section presents experimental results of our FPC-Curves modeling framework. We perform all our interactive curve design experiments on a PC with a 3.2 GHz Intel processor and 12 GB memory. Our method provides interpolating results at interactive speed, even for curves with a large number of input data points. In all our examples, the four types of control points: cusps, loops, inflection points and regular points are marked as purple triangles, green circle, blue stars and yellow squares, respectively. The end-points are marked as black squares.

Figures 1 and 8 show various curves with almost everywhere G^2 continuity constructed from the input data points in the second row. We can observe that the curves are visually pleasant and cusps and local loops only occur at the input data points, while the inflection points only appear at the input data points or joints.

As has been pointed out, we can generate curves with cusps, loops and inflection points by using previous interpolation methods. However, these methods are usually hard to create these

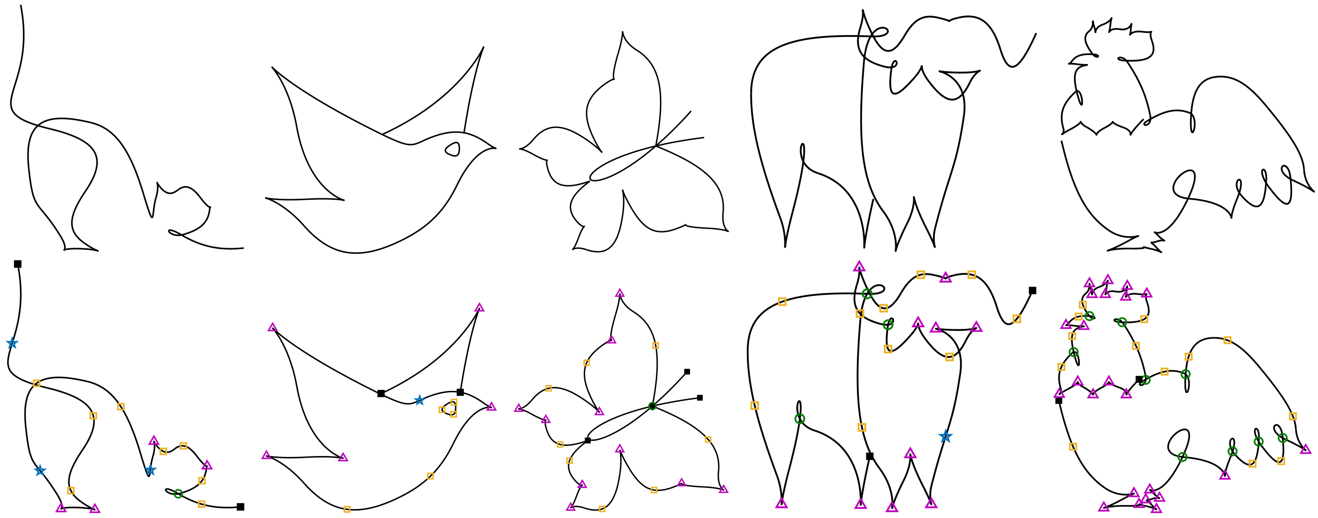

Figure 8: First row shows various FPC-Curves generated by our method from the interpolated points shown in the second row.

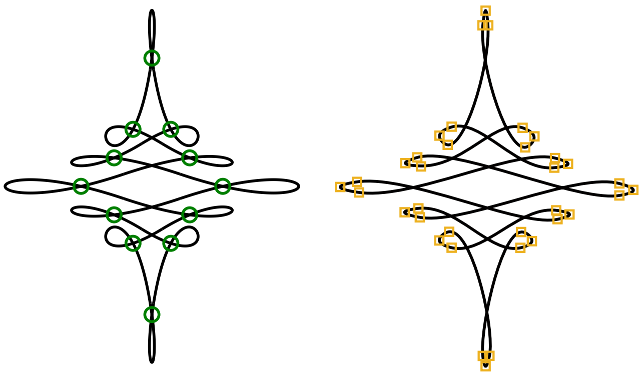

Figure 9: Comparison with the κ-Curves method [2]. To model a similar shape, our FPC-Curves method uses 12 points (left) designating the exact positions of the loops, while the κ-Curves method uses 36 points (right).

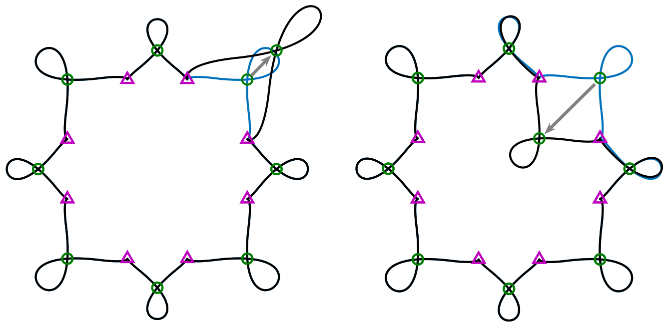

Figure 10: The shape of interpolatory FPC-Curves changes locally by relocating the associated interpolated points. The original curves are in blue, and the new curves are in black.

feature points at specified positions, or rely on user manipulation, or even introduce unexpected feature points. In addition, more control points are usually needed to model these features, introducing intense labor work for users. Figure 9 compares our FPC-Curves with the κ-Curves method [2], where the κ-Curves method uses 36 control points, while we use only 12 control points. Note that, in order to model a similar shape, we specify a total of 24 parameters ($t^*_{1,i}$ and $t^*_{2,i}$ for each control points) in addition to 12 control points in our method. For the other results in this paper, we just use the default values for $t^*_{1,i}$ and $t^*_{2,i}$.

While our algorithm generates a global solution, the influence of relocating an interpolated point drops dramatically when moving away from this point. Figure 10 shows an example curve where an interpolated point at the upper right corner is moved in opposite directions, where the original curve and the new curves are drawn in blue and black, respectively. From the results, we observe that relocating a control point only has a very local effect on the shape of the spline.

6. Conclusion and future work

Interpolation of a sequence of given data points is a widely used technique for shape modeling. Feature points such as cusps, loops and inflection points, which are considered to be artifacts in applications that require fairness of curves, are actually desired in artistic design, as these points describe the salient shape features. Previous interpolation methods suffer from the limitation of hard controlling over the location of feature points, sometimes even introduce unexpected feature points. To achieve a good control over the location and type of feature points, we use cubic parametric polynomial curves, where loops, cusps, and inflection points are mutually exclusive. FPC-Curves guarantee that cusps and local loops only occur at the interpolated points and the inflection points only occur at the interpolated points or joints. The resulting curves are guaranteed to be G^1 continuous and can achieve curvature continuous almost everywhere. Moreover, our curves have very good locality, i.e., relocating a control point only affects the local shape nearby the changed point. These properties make our method very suitable for artistic design applications.

Currently, we only focus on controlling over cusps, loops, and inflection points. The curvature extrema are also argued to be feature points of a shape. The discussion of curvature extrema of cubic curves have been discussed in [19]. Allowing control over curvature extrema will provide more flexibility in controlling over the curve shapes, which will be our future work.

Acknowledgements

The research of Zhonggui Chen and Juan Cao was supported by the National Natural Science Foundation of China (Nos. 61872308, 61472332, 61572020), the Natural Science Foundation of Fujian Province of China (No. 2018J01104), and the Fundamental Research Funds for the Central Universities (Nos. 20720190063, 20720190011). The research of Yongjie Jessica Zhang was supported in part by the PECASE award N00014-16-1-2254, NSF CAREER Award OCI-1149591, NSF CBET-1804929, CMU Manufacturing Futures Initiative and CMU-PITA.

Appendix A. Geometric conditions for interpolation

We give a more detailed discussion about geometric conditions for a cubic Bézier curve (2) interpolating a point at the parameter t^* such that the interpolated point becomes either a cusp, an inflection point, or a regular point. We prove that all the conditions can be expressed as linear combinations of the control points $\boldsymbol{P}_i$, $i = 1, 2, 3$. We first introduce an identity that will be employed by the following proofs, and we have

$$A\boldsymbol{P}_1 + 3C\boldsymbol{P}_3 = B\boldsymbol{P}_2. \tag{A.1}$$

This identity can be easily proved by expanding the determinants. Then we discuss each case respectively.

Appendix A.1. Condition (6) for cusps

A general cubic Bézier curve has a cusp if and only if $\Delta = B^2 - 4AC = 0$. If we specify the parameter of the cusp as $t^* \in (0, 1)$, then $t^* = -\frac{B}{2A} = -\frac{2C}{B}$, where $B \neq 0$. Thus, we have

$$2A = -\frac{B}{t^*}, \tag{A.2}$$

and

$$2C = -t^* B. \tag{A.3}$$

(A.2)$\times\boldsymbol{P}_1$ + (A.3)$\times 3\boldsymbol{P}_3$ yields:

$$2A\boldsymbol{P}_1 + 6C\boldsymbol{P}_3 = -3Bt^*\boldsymbol{P}_3 - \frac{B}{t^*}\boldsymbol{P}_1.$$

With Eq. (A.1), we have $2B\boldsymbol{P}_2 = -3Bt^*\boldsymbol{P}_3 - \frac{B}{t^*}\boldsymbol{P}_1$. Condition (6) can be obtained straightforwardly from the above equation.

Appendix A.2. Condition (11) for inflection points

The conditions for an interpolated point to be an inflection point are Eq. (10) and Eq. (9). Rewriting condition (10) as

$$B = -2A(t^* + h), \tag{A.4}$$

and substituting it into condition (9), we have

$$C = (t^{*2} + 2ht^*)A. \tag{A.5}$$

Thus, (A.5)$\times 3\boldsymbol{P}_3$− (A.4)$\times\boldsymbol{P}_2$ yields

$$3C\boldsymbol{P}_3 - B\boldsymbol{P}_2 = 3(t^{*2} + 2ht^*)A\boldsymbol{P}_3 + 2(t^* + h)A\boldsymbol{P}_2.$$

Condition (11) becomes an immediate consequence of the above equation using the identity (A.1).

Appendix A.3. Condition (14) for regular points

The conditions for an interpolated point to be a regular point are Eq. (A.6) and Eq. (13). Eq. (A.6) can be rewritten as

$$B = -2t^* A. \tag{A.6}$$

Substituting Eq. (A.6) into Eq. (13), we have

$$-C = (1 + 2t^*)A. \tag{A.7}$$

Thus, (A.7)$\times 3\boldsymbol{P}_3$ + (A.6)$\times\boldsymbol{P}_2$ yields

$$B\boldsymbol{P}_2 - 3C\boldsymbol{P}_3 = 3(1 + 2t^*)A\boldsymbol{P}_3 - 2t^* A\boldsymbol{P}_2.$$

Following the identity (A.1), we obtain condition (14) from the above equation.

Appendix B. Linear system of joints

We use an integer number $k_i \in \{0, 1, 2, 3\}$ to indicate the type of an interpolated point $\boldsymbol{v}_i$, where 0, 1, 2, and 3 represent a loop, a cusp, an inflection point, or a regular point, respectively. The coefficients of the linear system (19) are determined by parameters t_i^* ($t_{i,1}^*$ and $t_{i,2}^*$ for loops), λ_i, h_i, $\boldsymbol{v}_i$ and k_i as

$$\begin{cases} E_{i,1} &= M_{i,1}^{k_i}, \\ E_{i,2} &= M_{i,2}^{k_i} + \lambda_i M_{i+1,3}^{k_{i+1}}, \\ E_{i,3} &= \lambda_i M_{i+1,4}^{k_{i+1}}, \\ \boldsymbol{C}_i &= M_{i,5}^{k_i}\boldsymbol{v}_i + \lambda_i M_{i+1,6}^{k_{i+1}}\boldsymbol{v}_{i+1}, \end{cases}$$

where

$$\begin{cases} M_{i,1}^0 = \frac{(t_{i,1}^*-1)(t_{i,2}^*-1)}{t_{i,1}^* t_{i,2}^*}, \\ M_{i,2}^0 = -\frac{t_{i,1}^* t_{i,2}^* - 2t_{i,1}^* - 2t_{i,2}^* + 3}{(t_{i,1}^*-1)(t_{i,2}^*-1)}, \\ M_{i,3}^0 = -\frac{t_{i,1}^* t_{i,2}^* + t_{i,1}^* + t_{i,2}^*}{t_{i,1}^* t_{i,2}^*}, \\ M_{i,4}^0 = \frac{t_{i,1}^* t_{i,2}^*}{(t_{i,1}^*-1)(t_{i,2}^*-1)}, \\ M_{i,5}^0 = \frac{{t_{i,1}^*}^2 + {t_{i,2}^*}^2 + t_{i,1}^* t_{i,2}^* - 2t_{i,1}^* - 2t_{i,2}^* + 1}{t_{i,1}^* t_{i,2}^* (t_{i,1}^*-1)(t_{i,2}^*-1)}, \\ M_{i,6}^0 = \frac{{t_{i,1}^*}^2 + {t_{i,2}^*}^2 + t_{i,1}^* t_{i,2}^* - t_{i,1}^* - t_{i,2}^*}{t_{i,1}^* t_{i,2}^* (t_{i,1}^*-1)(t_{i,2}^*-1)}, \end{cases} \quad \begin{cases} M_{i,1}^1 = \frac{(t_i^*-1)^2}{t_i^{*2}}, \\ M_{i,2}^1 = -\frac{t_i^*-3}{t_i^*-1}, \\ M_{i,3}^1 = -\frac{t_i^*+2}{t_i^*}, \\ M_{i,4}^1 = \frac{t_i^{*2}}{(t_i^*-1)^2}, \\ M_{i,5}^1 = \frac{3t_i^*-1}{t_i^{*2}(t_i^*-1)}, \\ M_{i,6}^1 = \frac{3t_i^*-2}{t_i^*(t_i^*-1)^2}, \end{cases}$$

$$\begin{cases} M_{i,1}^2 = 1 - \frac{2t_i^{*2} + 6ht_i^* - 3t_i^* - 4h + 1}{t_i^*(t_i^{*2} + 4ht_i^* - 2h - t_i^*)}, \\ M_{i,2}^2 = \frac{2t_i^{*2} - 2t_i^* + 6ht_i^* - 4h}{(t_i^*-1)(t_i^{*2} + 4ht_i^* - 2h - t_i^*)} - 1, \\ M_{i,3}^2 = -\frac{2t_i^{*2} - 2t_i^* + 6ht_i^* - 2h}{t_i^*(t_i^{*2} + 4ht_i^* - 2h - t_i^*)} - 1, \\ M_{i,4}^2 = \frac{t_i^{*3} + 4ht_i^{*2}}{(t_i^*-1)(t_i^{*2} + 4ht_i^* - 2h - t_i^*)}, \\ M_{i,5}^2 = \frac{3t_i^{*2} + 6ht_i^* - 4t_i^* - 4h + 1}{t_i^*(t_i^*-1)(t_i^{*2} + 4ht_i^* - 2h - t_i^*)}, \\ M_{i,6}^2 = \frac{3t_i^{*2} + 6ht_i^* - 2t_i^* - 2h}{t_i^*(t_i^*-1)(t_i^{*2} + 4ht_i^* - 2h - t_i^*)}, \end{cases} \quad \begin{cases} M_{i,1}^3 = \frac{2t_i^{*4} + 4t_i^{*3} - 12t_i^{*2} + 4t_i^* + 2}{2t_i^{*4} + 5t_i^{*3} - 4t_i^{*2} - 3t_i^*}, \\ M_{i,2}^3 = -\frac{2t_i^{*4} + 4t_i^{*3} - 12t_i^{*2} - 6t_i^*}{2t_i^{*4} + 5t_i^{*3} - 4t_i^{*2} - 3t_i^*}, \\ M_{i,3}^3 = -\frac{2t_i^{*4} + 6t_i^{*3} + 3t_i^{*2} - 8t_i^* - 3}{2t_i^{*4} + 5t_i^{*3} - 4t_i^{*2} - 3t_i^*}, \\ M_{i,4}^3 = \frac{2t_i^{*4} + 6t_i^{*3} + 3t_i^{*2}}{2t_i^{*4} + 5t_i^{*3} - 4t_i^{*2} - 3t_i^*}, \\ M_{i,5}^3 = \frac{2 + 10t_i^*}{2t_i^{*4} + 5t_i^{*3} - 4t_i^{*2} - 3t_i^*}, \\ M_{i,6}^3 = \frac{3 + 8t_i^*}{2t_i^{*4} + 5t_i^{*3} - 4t_i^{*2} - 3t_i^*}. \end{cases}$$

References

[1] G. Farin, Curves and surfaces for CAGD: a practical guide, Morgan Kaufmann, 2002.

[2] Z. Yan, S. Schiller, G. Wilensky, N. Carr, S. Schaefer, κ-Curves: interpolation at local maximum curvature, ACM Transactions on Graphics 36 (4) (2017) 129:1–129:7.

[3] R. L. Levien, From spiral to spline: optimal techniques in interactive curve design, Ph.D. thesis, University of California, Berkeley, USA (2009).

[4] G. Brunnett, J. Kiefer, Interpolation with minimal-energy splines, Computer-Aided Design 26 (2) (1994) 137–144.

[5] R. Levien, C. H. Séquin, Interpolating splines: which is the fairest of them all?, Computer-Aided Design and Applications 6 (1) (2009) 91–102.

[6] J. Peters, Chapter 8 - geometric continuity, in: G. Farin, J. Hoschek, M.-S. Kim (Eds.), Handbook of Computer Aided Geometric Design, North-Holland, Amsterdam, 2002, pp. 193–227.

[7] Y. Y. Feng, J. Kozak, On G^2 continuous interpolatory composite quadratic Bézier curves, Journal of Computational and Applied Mathematics 72 (1) (1996) 141–160.

[8] E. Catmull, R. Rom, A class of local interpolating splines, Computer Aided Geometric Design (1974) 317–326.

[9] G. Deslauriers, S. Dubuc, Symmetric iterative interpolation processes, Constructive Approximation 5 (1) (1989) 49–68.

[10] H. P. Moreton, Minimum curvature variation curves, networks, and surfaces for fair free-form shape design, Ph.D. thesis, University of California at Berkeley, Berkeley, CA, USA (1992).

[11] G. Farin, Degree reduction fairing of cubic B-spline curves, in: Geometry Processing for Design and Manufacturing, 1992, pp. 87–99.

[12] G. Harary, A. Tal, The natural 3D spiral, Computer Graphics Forum 30 (2) (2011) 237–246.

[13] D. Walton, D. Meek, A planar cubic Bézier spiral, Journal of Computational and Applied Mathematics 72 (1) (1996) 85–100.

[14] G. Farin, Class A Bézier curves, Computer Aided Geometric Design 23 (7) (2006) 573–581.

[15] J. Cao, G. Wang, A note on class A Bézier curves, Computer Aided Geometric Design 25 (7) (2008) 523–528.

[16] C. Y. Wang, Shape classification of the parametric cubic curve and parametric B-spline cubic curve, Computer-Aided Design 13 (4) (1981) 199–206.

[17] B.-Q. Su, D.-Y. Liu, An affine invariant and its application in computational geometry, Scientia Sinica (Series A) 24 (3) (1983) 259–267.

[18] M. C. Stone, T. D. DeRose, A geometric characterization of parametric cubic curves, ACM Transactions on Graphics 8 (3) (1989) 147–163.

[19] D. Walton, D. Meek, Curvature extrema of planar parametric polynomial cubic curves, Journal of Computational and Applied Mathematics 134 (1) (2001) 69–83.