Circle Reproduction with Interpolatory Curves at Local Maximal Curvature Points

Zhipei Yan[a,*], Stephen Schiller, Scott Schaefer[a]

[a] *Texas A&M University*

Abstract

We present a piece-wise rational, quadratic, interpolatory curve that is able to reproduce circles and other elliptical or hyperbolic shapes. The curve is curvature continuous except at infection points and points of local maximum curvature appear only at control points and nowhere else. The local maximum curvature property ensures that users have direct control over salient points of the curve, and users can control if and where features such as cusps and loops are generated.

To construct the desired curve, we formulated an energy that encodes the desired properties to optimize using a boxed constrained optimization. We provide an efficient algorithm for choosing an initial guess close to the solution to accelerate convergence. In addition, we show how to automatically choose the rational weights of the curve as part of the optimization to reproduce shapes such as circles.

Keywords: interpolation, spline curves, κ-Curves, curvature, circle reproduction
2010 MSC: 68U05, 65D17

1. Introduction

Interpolatory curves have a long history in curve modeling. As opposed to approximating representations, interpolatory curves pass through a set of control points specified by the user. While initially such control seems desirable, interpolatory curves have been plagued by shape problems such as loops and cusps that can appear away from control points. Hence, it can be difficult for the user to control the placement of these features or even whether or not cusps and loops should exist within the curve.

Yan et al [1] showed that control points should be located at features of the curve. Moreover, such features should not occur in the curve outside of the control points. Levien [2] pointed out that human beings are more sensitive

*Corresponding author
Email address: yanzp@tamu.edu (Zhipei Yan)

to minima and maxima of curvature. Hence, it makes sense to identify local maxima of curvature as features of the curve and require that local maxima of curvature appear only at control points.

Beyond simply controlling features of the curve, the types of curves a modeling tool can produce are also important. While polynomial curves are prevalent due to their ease of use, circles are important shapes due to their common occurrence in every day life whether part of images or cylindrical or spherical 3D shapes. Therefore, a useful curve representation would be able to represent shapes such as circles exactly. In particular, if the interpolated control points lie on a circle, even if they are not equally spaced, it would be reasonable to expect that a circle would be reproduced. When the control points do not lie on a circle, the curve should be "fair" where "fair" means the curvature plot should be simple [3].

For curve editing applications, it is also important that the curve moves continuously with movement of the control points. This requirement is satisfied by many interpolatory curves, but not all. Clothoid splines [4], for example, may suddenly change shape as their control points are dragged, introducing a loop where there was none before [1].

In this paper, we introduce a piece-wise rational curve that interpolates a series of input points on a 2D plane. All local maxima of curvature only occur at input points. For each input point, we compute a single rational quadratic curve that interpolates the input point somewhere in its domain. We use the remaining degrees of freedom in the curve to connect consecutive curves with curvature continuity everywhere except at inflections. At these inflection points, the magnitude of curvature will be continuous though the sign (positive/negative) is different. We build a box constrained optimization to solve for the final curve and use an iterative method to compute a close initial guess to accelerate the optimization. Finally, we show how to automatically compute the rational weights of the curve and incorporate this function into the optimization to ensure that circles are reproduced when the control points lie on a circle.

2. Related work

Piecewise polynomial and rational functions are commonly used to represent interpolatory curves. There are many smooth interpolatory curve constructions ranging from hermite curves [5], Catmull-Rom curves [6], and interpolatory subdivision curves [7]. While these constructions are easy to create and smooth, these methods enforce only parametric properties rather than geometric properties such as curvature in their construction.

Beyond parametric continuity, researchers have studied the conditions for geometric continuity as well. Schaback [8] describes how to create G^2 connected quadratic Bézier curves to interpolate a series of non-inflecting points on a 2D plane, which means the sign of the curvature of the curve should be always positive or always negative. Feng [9] solves a similar problem and creates a series of quadratic Bézier curves, each of which interpolates a control point on the interior of the parameter range for the curve while remaining G^2 at the

join points for the curve. Like Schaback, Feng's approach is restricted to non-inflecting points, though their numerical solver can be applied on an "S" shape.

κ-Curves [1] utilize Feng's approach to interpolate a series of points at local maximum curvature points on a 2D plane. The quadratic Bézier curve components are parabolas and all have a unique max curvature point. κ-Curves enforce a local maximum curvature at all input points. However, the authors relax the G^2 condition at join points (where two Bézier curves meet) that form inflection points. Instead the authors require that the absolute value of curvature is continuous, which means that κ-Curves can reproduce curves with inflection points. The authors enumerate the desired geometric criteria and use an iterative method to create a curve with those properties. Unfortunately, the iterative method does not guarantee convergence of the result, though the method appears to work well in practice. In addition, the polynomial nature of their representation means that common shapes such as circles cannot be reproduced.

Arc or circular splines are widely used to represent circles. Hoschek [10] inserts an arc of a circle between each pair of adjacent control points and connect all arcs with G^1 continuity at the control points. Meek [11], Yeung [12], and Kurnosenko [13] utilize biarcs, two circular arcs connected using G^1 point, between two control points and interpolate given tangent vectors at the control points where the curve is G^1. Meek [14] uses C-Shape curves of an arc and a conic to interpolate points, tangent vectors, and curvatures. Piegl [15] developed a method to estimate the tangent directions at control points and then interpolate data using biarcs. While these methods can obviously reproduce circles, the curves lack curvature continuity and do not allow the user to control points of local maximum curvature.

Many non-polynomial methods have been developed to represent circles as well. Wenz [16], Sequin [17] and Sun [18] compute local circle arcs using three adjacent control points and then blend each pair of the adjacent circles to obtain a C^2 spline though the authors do not control the placement of curvature maxima. Schaefer [19] creates interpolatory curves through subdivision that can reproduce circles, though the method is sensitive to the parametric spacing of points along a circle.

Rational quadratic polynomial curves form conic sections and have the capability to represent circles exactly. Xu [20] and Canton [21] researched the geometric properties of conic sections in rational Bézier form. Similar to [8], [22] explored the existence and properties of interpolating a set of non-inflecting points using G^2 connected conic splines. Yang [23] compute a G^1 quadratic interpolatory splines and then adjust the tangent vectors at control points and the weights of the Bézier control points to achieve a G^2 curve. In this paper, we use G^2 almost everywhere connected conic splines to interpolate a series of 2D points with local max curvature points only occur at the interpolated points. Unlike many papers, our input points do not need to be non-inflecting. At inflection points, we maintain continuity in the magnitude of curvature, though the sign of curvature inverts.

3. Spline Representation

To represent our curve, we use rational quadratic Bézier curves as our curve primitive: one curve $c_i(t)$ for each interpolated input point $\{Q_i\}$. We represent each curve $c_i(t)$ in *standard form* [5] defined by three control points $\{P_{i,0}, P_{i,1}, P_{i,2}\}$ with positive weight w_i for $P_{i,1}$ and unit weights for the remaining control points:

$$c_i(t) = \frac{(1-t)^2 P_{i,0} + 2(1-t)t w_i P_{i,1} + t^2 P_{i,2}}{(1-t)^2 + 2(1-t)t w_i + t^2}, \quad t \in [0,1], \quad w_i > 0 \tag{1}$$

with the constraint that there exists some $t_i \in (0,1)$ such that

$$c_i(t_i) = Q_i, \tag{2}$$

and Q_i is the local max curvature point of $c_i(t)$. We choose rational quadratics for their ability to represent shapes such as circles. Furthermore, the curvature profile for these curves are simple compared to higher degree curves. For $w_i >$ 1, the curves form one branch of a hyperbola and has at most one maximal curvature point. When $w_i = 1$, the curve is a parabola and, again, has at most one maximal curvature point. Finally, when $w_i < 1$, the curve is an arc of an ellipse and may have at most one maximal curvature point and one minimal curvature point [24].

3.1. Curvature

Given the importance of maximal curvature points in our construction, we derive the t-values for the curvature extrema of rational Bézier curves in this section. The curvature $\kappa_i(t)$ of 2D curve $c_i(t)$ is

$$\kappa_i(t) = \frac{\det(c_i'(t), c_i''(t))}{\sqrt{c_i'(t) \cdot c_i'(t)}^3}, \tag{3}$$

and the local extreme curvature points are the roots of $\kappa_i'(t) = 0$. For our curves, $\kappa_i'(t)$ is a ratio between a polynomial and the square root of another polynomial in the variable t. The denominator of this expression depends solely on the length of the tangent $c_i'(t)$, which is non-negative everywhere. Hence, we need only consider the numerator of $\kappa_i'(t)$ when analyzing its roots where the the numerator is

$$\det(c_i'(t), c_i''(t))'(c_i'(t) \cdot c_i'(t)) - \frac{3}{2}\det(c_i'(t), c_i''(t))(c_i'(t) \cdot c_i'(t))'$$

This expression is a quartic polynomial in t. When $w_i = 1$, the leading coefficient of the numerator is zero and the expression degenerates to a cubic polynomial. Rewriting the numerator in the Bézier basis yields

$$\begin{bmatrix} (1-t)^4 \\ 4(1-t)^3 t \\ 6(1-t)^2 t^2 \\ 4(1-t)t^3 \\ t^4 \end{bmatrix}^T \cdot \begin{bmatrix} -2w_i^3|P_{i,0} - P_{i,1}|^2 + w_i(P_{i,0} - P_{i,1}) \cdot (P_{i,0} - P_{i,2}) \\ -w_i^2|P_{i,0} - P_{i,1}|^2 + \frac{1}{4}|P_{i,0} - P_{i,2}|^2 \\ \frac{1}{2}w_i(P_{i,2} - P_{i,0}) \cdot (P_{i,0} - 2P_{i,1} + P_{i,2}) \\ -\frac{1}{4}|P_{i,0} - P_{i,2}|^2 + w_i^2|P_{i,1} - P_{i,2}|^2 \\ -w_i(P_{i,0} - P_{i,2}) \cdot (P_{i,1} - P_{i,2}) + 2w_i^3|P_{i,1} - P_{i,2}|^2 \end{bmatrix}. \tag{4}$$

We care about the real roots of Equation (4) in interval $[0, 1]$. A general quartic function has up to 4 real roots. Equation (4) may have 0, 1 or 2 roots in range [0,1] depending on the parameter w_i, which controls whether the curve is piece of a hyperbola, parabola, or ellipse. The first two shapes can only have zero or one critical points of curvature in the interval $(0, 1)$. In the elliptical case, Equation 4 can have up to four roots corresponding to the two maximal and two minimal points of curvature on an ellipse, but at most two of these roots, corresponding to one maximal and one minimal curvature point, can fall into the $(0, 1)$ interval. Figure 1 shows each of these cases.

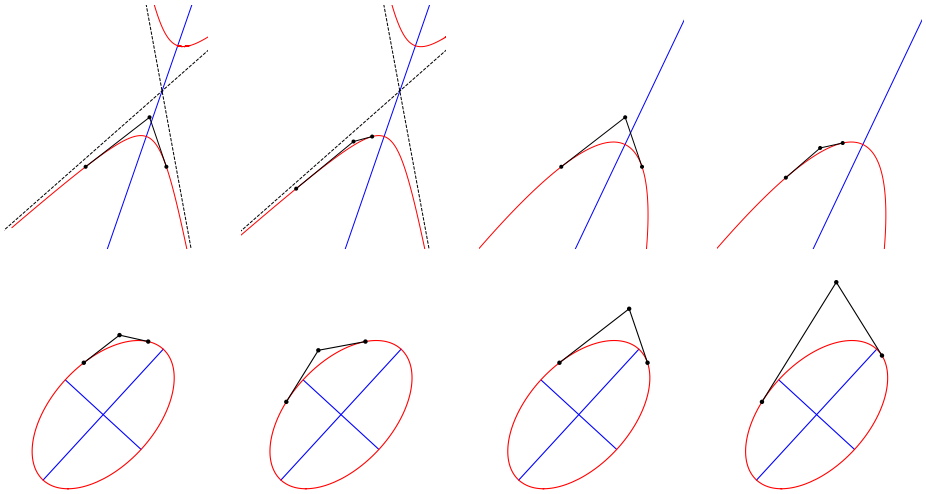

Figure 1: Rational Bézier curves may have 0, 1 or 2 critical points of curvature within the parameter range (0,1). Solid black lines are Bézier control polygons, red lines are rational Bézier curves, blue lines are axes of symmetry for the conic sections. The first row shows a hyperbola (first two images) and a parabola (last two images), each of which can have one or zero max curvature points within a rational Bézier curve. The last row shows an ellipse, which may have 0, 1 min, 1 max, or 1 min and 1 max curvature points.

To build a curvature continuous curve, we also need the value of curvature at the end-points of each rational quadratic curve. Setting t to 0 or 1 in Equation (3) yields the curvature at the two end points:

$$\begin{aligned} \kappa_i(0) &= \frac{1}{w_i^2} \frac{\Delta(P_{i,0}, P_{i,1}, P_{i,2})}{||P_{i,1} - P_{i,0}||^3} \\ \kappa_i(1) &= \frac{1}{w_i^2} \frac{\Delta(P_{i,0}, P_{i,1}, P_{i,2})}{||P_{i,2} - P_{i,1}||^3} \end{aligned} \tag{5}$$

where $\Delta(\cdot, \cdot, \cdot)$ is the signed area of a triangle.

3.2. Interpolation at Critical Curvature Points

For each curve segment $c_i(t)$, we want the input point Q_i to be located at a local maximum of curvature. For a single curve, we can do so by rewriting $c_i(t)$

in a Lagrange-type basis in terms of three interpolated points: $P_{i,0}$, Q_i, and $P_{i,2}$. Since we know the parameters associated with the interpolated end-points (i.e.; $c_i(0) = P_{i,0}$ and $c_i(1) = P_{i,2}$), we need only find the parameter t_i such that Q_i is at the point of maximum curvature. Solving for $P_{i,1}$ in Equation 2 yields

$$P_{i,1} = \frac{((1-t_i)^2 + 2(1-t_i)t_i w_i + t_i^2)Q_i - (1-t_i)^2 P_{i,0} - t_i^2 P_{i,2}}{2(1-t_i)t_i w_i}. \quad (6)$$

Substituting Equation 6 into Equation 4 gives a simple, polynomial expression for t_i in terms of the three interpolated points

$$-w_i A_i (1-t_i)^4 - (A_i + B_i)(1-t_i)^3 t_i + (B_i + C_i)(1-t_i)t_i^3 + w_i C_i t_i^4 = 0 \quad (7)$$

where

$$\begin{aligned} A_i &= (P_{i,0} - Q_i) \cdot (P_{i,0} - Q_i) \\ B_i &= (P_{i,0} - Q_i) \cdot (P_{i,2} - Q_i) \\ C_i &= (P_{i,2} - Q_i) \cdot (P_{i,2} - Q_i). \end{aligned}$$

Though a general quartic polynomial may have multiple or no roots in range $[0, 1]$, we prove that Equation 7 always has a unique root in [0,1] in Appendix A. Notice here we only solved the root for $\kappa'(t)$=0, which means the interpolated point Q_i is either a local minimum or a maximum point. If a small weight $w_i < 1$ is chosen for point $P_{i,1}$, Q_i may be a local minimum of $\kappa(t)$. However, in practice, our choice of weight function and optimization procedure converges to a local maximum point. When $w_i \geq 1$, all interpolated points will be local maximum points of $\kappa(t)$.

4. Curve Construction

Our strategy to construct a curve satisfying our interpolation, curvature continuity, and maximal curvature constraints is to form an energy encoding all of these constraints. While the energy is nonlinear, which may make finding its minimum difficult, we will provide a method for choosing an initial guess close to the minimum in Section 5.1. From there, our optimization can quickly converge to the minimal result. Note that this is a different strategy than [1] used for non-rational curves. Instead, the authors provide an alternating algorithm that performs well in practice but does not guarantee that a minimum is actually reached. While this section assumes that the curve is closed, we will relax this assumption to handle curves with boundaries in Section 6.

4.1. Interpolation constraints

While we use an energy for the majority of our geometric properties, we encode the interpolation condition as a constraint. Compared with the other terms, interpolation is a simple expression that involves only a few variables.

Since we allocate one curve per input point, we can combine Equations 1 and 2 to produce a set of polynomial constraints:

$$\begin{cases} \vdots \\ (1-t_i)^2(P_{i,0}-Q_i) + 2(1-t_i)t_i w_i(P_{i,1}-Q_i) + t_i^2(P_{i,2}-Q_i) = 0 \\ \vdots \end{cases} \tag{8}$$

4.2. Smoothness

To create curvature continuity between consecutive curves $c_{i-1}(t)$ and $c_i(t)$, we must enforce C^0 and G^1 continuity in addition to curvature continuity. C^0 continuity is trivial and simply requires $P_{i-1,2} = P_{i,0}$. G^1 continuity requires that the tangents of each curve point in the same direction where the two curves meet. Substituting the C^0 condition into this requirement yields the expression

$$P_{i-1,2} = P_{i,0} = (1-\lambda_i)P_{i-1,1} + \lambda_i P_{i,1} \tag{9}$$

where $\lambda_i \in (0,1)$. We utilize the C^0 and G^1 conditions as constraints and substitute Equation 9 into Equation 8 to eliminate the variables $P_{i,0}$ and $P_{i,2}$ from the optimization with new variables λ_i.

Curvature continuity requires the curvatures at the end-points of consecutive curves to be equal. As noted by [1], requiring curvature continuity for quadratic curves at an inflection point is impossible since quadratics cannot possess zero curvature unless the entire curve degenerates to a line. Rational quadratics possess these same property. Hence, we cannot require that curves are curvature continuous everywhere. Instead, like [1], we require that the absolute value of curvature is continuous, meaning that the magnitude of curvature is continuous at inflection points but the sign of curvature will be discontinuous. We can easily encode this requirement as an error function of the squared curvature using Equation 3.

$$(k_{i-1}(1)^2 - k_i(0)^2)^2 = \left(\frac{\Delta(P_{i-1,0}, P_{i-1,1}, P_{i-1,2})^2}{w_{i-1}^4 ||P_{i-1,1} - P_{i-1,0}||^6} - \frac{\Delta(P_{i,0}, P_{i,1}, P_{i,2})^2}{w_i^4 ||P_{i,2} - P_{i,1}||^6} \right)^2$$

Combining this expression with Equation 9 gives

$$\frac{1}{||P_{i,1} - P_{i-1,1}||^{12}} \left(\frac{\Delta(P_{i-2,1}, P_{i-1,1}, P_{i,1})^2 (1-\lambda_{i-1})^2}{w_{i-1}^4 \lambda_i^4} - \frac{\Delta(P_{i-1,1}, P_{i,1}, P_{i+1,1})^2 \lambda_{i+1}^2}{w_i^4 (1-\lambda_i)^4} \right)^2 \tag{10}$$

Given that $||P_{i,1} - P_{i-1,1}|| > 0$, dropping this term does not affect the nullspace of this energy term. Eliminating this term gives our final energy

$$E_{G^2}(\{P_{i,1}, \lambda_i\}) = \sum_{i=1}^{n} \left(\frac{\Delta(P_{i-2,1}, P_{i-1,1}, P_{i,1})^2 (1-\lambda_{i-1})^2}{w_{i-1}^4 \lambda_i^4} - \frac{\Delta(P_{i-1,1}, P_{i,1}, P_{i+1,1})^2 \lambda_{i+1}^2}{w_i^4 (1-\lambda_i)^4} \right)^2. \tag{11}$$

4.3. Maximum Curvature

For each curve segment $c_i(t)$, Equation 7 should be satisfied to ensure Q_i is interpolated at local max curvature point. We utilize the L_2 norm of Equation 7 and normalize the expression by dividing by the leading coefficient of the polynomial. This process yields the energy term for local maximal curvature

$$\begin{aligned} &E_c(\{P_{i,j}, t_i\}) \\ &= \sum_{i=0}^{n} \left(\frac{-w_i A_i (1-t_i)^4 - (A_i + B_i)(1-t_i)^3 t_i + (B_i + C_i)(1-t_i) t_i^3 + w_i C_i t_i^4}{L_i}\right)^2 \end{aligned} \tag{12}$$

where L_i is the leading coefficient of the numerator in equation (12).

$$L_i = \begin{cases} (w_i - 1)(C_i - A_i), & w_i \neq 1 \\ A_i - 2B_i + C_i, & w_i = 1 \end{cases}$$

If we substitute Equation 9 into Equation 12, we can write E_c solely in terms of the variables $P_{i,1}, \lambda_i, t_i$.

4.4. Rational Weights

Up until this point, we have assumed that the rational weight w_i has been fixed. w_i provides the user additional control over the shape of the curve and can be used to adjust the shape of the curve. However, for users who do not wish to manually adjust w_i, we can choose w_i automatically to optimize for certain geometric properties of the curve. Our motivation is to choose w_i to produce a circle when the user places input points Q_i along a circle. Unlike other methods [19], we do not require that the vertices are evenly spaced in terms of arc length along the circle to reproduce this shape.

To choose w_i we write w_i as a function $\tilde{w}_i$ of $\{P_{i,1}, \lambda_i\}$. We then write an energy term that measures the deviation of w_i from these weights as

$$E_w(P_{i,1}, \lambda_i, w_i) = \sum_{i=1}^{n} (w_i - \tilde{w}_i(P_{i,1}, \lambda_i))^2 \tag{13}$$

4.4.1. Minimum Eccentricity Weights

All rational quadratic curves define a conic section as the solution to an implicit quadratic equation

$$ax^2 + bxy + cy^2 + dx + ey + f = 0. \tag{14}$$

We can easily transform our parametric quadratics into this implicit form using a resultant [25]. To create circles, we use the eccentricity of this conic, which measures the deviation of the shape from a circle [26]. For circles, the eccentricity is 0. Ellipses have an eccentricity between 0 and 1. Parabolas have eccentricity 1. And hyperbolas have eccentricity greater than 1. The eccentricity for conics of the form in Equation 14 is

$$\epsilon = \sqrt{\frac{2\sqrt{(a-c)^2 + b^2}}{a + c + \sqrt{(a-c)^2 + b^2}}}.$$

Our strategy is to choose w_i that minimizes the eccentricity squared ϵ^2. Substituting the coefficients of the implicit form of Equation 1 into the definition of ϵ^2, taking its derivative with respect to w_i, and solving for the critical point with respect to w_i yields a very simple expression for the rational weight [20]

$$\hat{w}_i = \sqrt{\frac{|P_{i,2} - P_{i,0}|^2}{2(|P_{i,0} - P_{i,1}|^2 + |P_{i,2} - P_{i,1}|^2)}}. \tag{15}$$

This expression is less than 1 for points in general position and, therefore, leads to elliptical arcs.

Figure 2 show an example of a single rational Bézier curve with three fixed control points. This figure shows the entire conic section instead of just the portion of the curve the corresponds to the Bézier curve with its parameter in $[0, 1]$. The only difference between the three curves shown is the rational weight w_i. We show two curves with fixed weights of 0.3 and 0.7 in blue and the curve with minimum eccentricity weight of 0.593 in red. Note that it is not possible to generate circle from this fixed set of control points by only manipulating w_i.

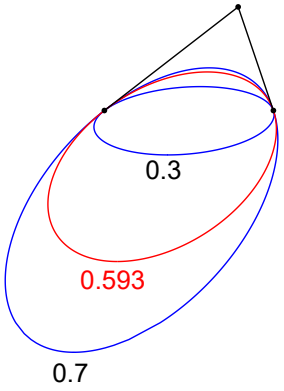

Figure 2: A rational Bézier curve with unit weights for the two end-points and different ellipses generated by modifying the weight of the central control point, which is set to 0.3 (blue), 0.7 (blue), and 0.593 (red), which is the minimal eccentricity weight for these control points.

Unfortunately, these minimum eccentricity weights cannot be directly used in our optimization. Equation 15 approaches zero when $P_{i,0}$ approaches $P_{i,2}$. Though a zero weight is valid for rational Bézier curves, Equation 6 becomes undefined, which leads to instability in the subsequent optimization for some shapes. Looking at this case from a geometric viewpoint, the control points of the Bézier curve are collapsing to form a line and potentially flipping orientation, which should change the sign of the curvature along the curve. This scenario corresponds to a cusp in the curve. Therefore, reproduction of a circle is undesirable as the curve at its point of maximum curvature should have a curvature value that approaches ∞.

Our solution to this issue is to simply clamp the minimum value of a weight to 0.5 to avoid these instabilities. Doing so means that we cannot produce a

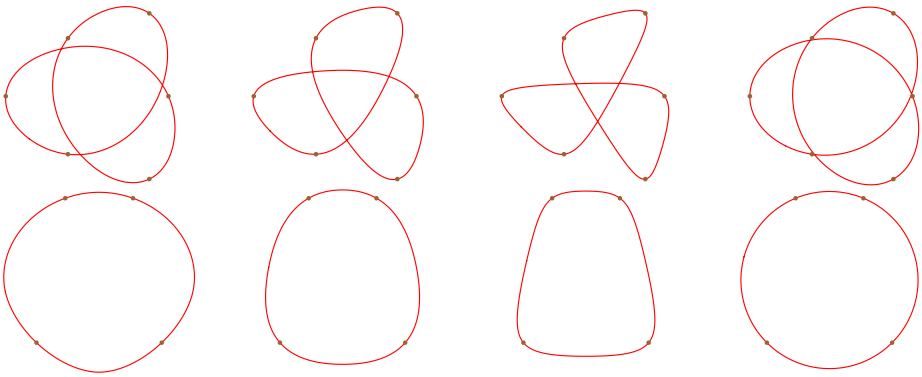

Figure 3: Each row contains the same input points Q_i but uses different weights for the input points in each column. The weights from left to right are: 0.6, 1, 2, and our clamped min eccentricity weight.

circle if the arc length spanned by the end-points of a Bézier curve is more than $\frac{1}{3}$ of the circle's perimeter. For a set of input points consisting of three points, we will only be able to produce a circle from an equilateral triangle. However, for other curves consisting of more control points, most non-uniform distributions of input points on a circle can reproduce that circle. Figure 3 shows an example of four, non-uniformly spaced points on a circle and the curves produced using different weights.

With this modification, our weight function w_i becomes

$$\tilde{w_i} = \begin{cases} \mu_i \hat{w}_i & \mu_i \hat{w}_i > \frac{1}{2} \\ \frac{1}{2} & \text{otherwise} \end{cases} \tag{16}$$

The coefficient μ_i in Equation 16 provides a form of tension control for the user to control the shape of the curve with the initial μ_i set to 1. The user can adjust this value to make the curve more round by reducing μ_i or more pointed by increasing μ_i. Figure 8 shows an example of modifying μ_i to affect the shape of the curve.

5. Optimization

Given the input points $\{Q_i\}_{i=1}^n$, we use the energy terms from previous sections to write an optimization problem for our curve as

$$\min_{P_{i,1} \in \mathrm{R}^2, \lambda_i \in (0,1), t_i \in (0,1), w_i > 0} E_{G^2}(P_{i,1}, \lambda_i) + E_c(P_{i,1}, \lambda_i, t_i) + E_w(P_{i,1}, \lambda_i, t_i, w_i) \tag{17}$$

with the constraints

$$\begin{cases} \vdots \\ (1-t_i)^2((1-\lambda_i)P_{i-1,1}+\lambda_i P_{i,1}-Q_i)+2(1-t_i)t_i w_i(P_{i,1}-Q_i) \\ +t_i^2((1-\lambda_{i+1})P_{i,1}+\lambda_{i+1}P_{i+1,1}-Q_i)=0 \\ \vdots \end{cases}, \quad (18)$$

which are formed by substituting Equation (9) into Equation (8). Note that E_w can be omitted if we do not desire automatic reproduction of circles, which also removes w_i from the variables but still gives the user control over the w_i to affect the shape of the curve.

Since the constraints (18) are linear in the points $\{P_{i,1}\}$, we can solve for the $\{P_{i,1}\}$ as a function of $\{\lambda_i, t_i, w_i\}$ from (18). Then insert the expressions of $P_{i,1}$ back to (17) and generate a new optimization with reduced number of variable:

$$\min_{\lambda_i,t_i,w_i} E_{G^2}(\lambda_i,w_i,t_i)+E_c(\lambda_i,w_i,t_i)+E_w(\lambda_i,w_i,t_i). \quad (19)$$

The constraints on the variables are now simplified to $\lambda_i \in (0,1), t_i \in (0,1)$ and $w_i > 0$. We use the Eigen library for solving linear system of $\{P_{i,1}\}$ and use the Alglib library for boxed constrained numerical optimization (19).

5.1. Initial guess

While any initial guess for the optimization could be chosen, we start with a very simple initial guess with $\lambda_i = 0.5$, $t_i = 0.5$, and $w_i = 1$. Though Section 5 optimizes the reduced form of the energy with the $P_{i,1}$, we use the $P_{i,1}$ as part of our iterative strategy for forming the initial guess. Using the values of λ_i, t_i, w_i, we solve for the $P_{i,1}$ from Equation (18), which produces an interpolatory C^1 curve. Yet this curve almost certainly does not satisfy the automatic weight conditions, curvature continuity, or maximum curvature conditions.

In most cases, this simple initial guess can lead to slow convergence. To accelerate the optimization, we propose a modification of the optimization from [1]. In that paper, the authors provide an iterative method to find their polynomial curve. However, the authors give no proof of convergence of this procedure nor do they show any energy function that their iterative method optimizes.

We provide a similar, iterative method for refining the initial guess before optimization. We iteratively solve for different variables by minimizing each term in Equation (17) in turn. Such a strategy is not guaranteed to minimize the total energy in Equation (17) but is only designed to create a better initial guess for the nonlinear optimization. In practice, this procedure rapidly refines the initial guess to a curve close to the minimum of our energy, which is then refined by the optimization procedure.

Starting from the current values of λ_i, t_i, w_i, we first hold all variables constant except for w_i and solve Equation (13) to make $E_w = 0$. Next we solve for t_i by holding the $\lambda_i, w_i, P_{i,1}$ to make $E_c = 0$. Hence, each t_i is defined by the

root of the quartic polynomial from Equation (7). Our next step is to hold all variables constant except for the λ_i and solve for λ_i from Equation (10) to make $E_{G^2} = 0$. Finally we solve the linear system in Equation (18) to update the $P_{i,1}$. We iterate this entire process several times.

This procedure projects the current solution onto each energy term in turn along the direction of only one set of variables. While there is no guarantee that the solution converges, we observe rapid convergence towards the minimizer for the early iterations with progress slowing afterwards. Hence, we only use this procedure to find a close initial guess for the optimization in Section 5, which then converges rapidly to the solution. Algorithm 1 provides pseudo code for the entire optimization procedure. Note that, in an interactive system, we can take advantage of the temporal coherence of prior solutions to further accelerate the optimization. To do so, we use the results from the previous optimization as the initial guess for the procedure instead of the default values for λ_i, t_i, and w_i.

Algorithm 1 Rational Curve

Input: $\{Q_i\}, \{\mu_i\}$
procedure INIT GUESS
    $\{\lambda_i\} \leftarrow 0.5$
    $\{t_i\} \leftarrow 0.5$
    $\{w_i\} \leftarrow 1$
    $\{P_{i,1}\} \leftarrow$ linearSolve Eq.(18)$(\{\lambda_i, t_i, w_i\})$
    for $j \leftarrow 1$ to 60 **do**
        $\{w_i\} \leftarrow \tilde{w_i}(P_{i,1}, \lambda_i)$
        $\{t_i\} \leftarrow$ quartic root of Eq. (7)
        $\{\lambda_i\} \leftarrow$ solve from Eq. (10)
        $\{P_{i,1}\} \leftarrow$ linearSolve Eq.(18)$(\{\lambda_i, t_i, w_i\})$
    return $\{\lambda_i, t_i, w_i\}$
procedure ENERGY(λ_i, t_i, w_i)
    $\{P_{i,1}\} \leftarrow$ linearSolve Eq.(18)$(\{\lambda_i, t_i, w_i\})$
    return $E_{G^2}(P_{i,1}, \lambda_i) + E_c(P_{i,1}, \lambda_i, t_i) + E_w(P_{i,1}\lambda_i, w_i)$
procedure MAIN
    $\{\lambda_i, t_i, w_i\} \leftarrow$ InitGuess()
    $\{\lambda_i, t_i, w_i\} \leftarrow$ minimize Energy(λ_i, t_i, w_i) with $0 < \lambda_i, t_i < 1$, $w_i > 0$
    $\{P_{i,1}\} \leftarrow$ linearSolve Eq.(18)$(\{\lambda_i, t_i, w_i\})$
    Output the control polygons using $\{P_{i,1}, \lambda_i, w_i\}$
    return

6. Curves with Boundary

Our algorithm is also applicable to open curves. The only difference is that we do not treat points cyclically and add two end points.

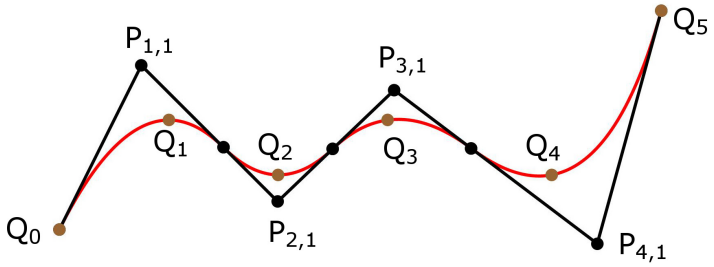

Figure 4: An open curve with 6 input points consists of 4 rational Bézier curve segments. The interior points $Q_{i=1\ldots4}$ are local max curvature points while Q_0 and Q_5 form the end points.

Given $n+2$ input points $\{Q_0, Q_1, \cdots, Q_{n+1}\}$ and n weights $\{w_i\}_{i=1}^n$, we compute the parameters $\{\lambda_i\}_{i=2}^n$, $\{t_i\}_{i=1}^n$ and n off-curve points $\{P_{i,1}\}_{i=1}^n$ to generate n rational Bézier curve segments $\{c_i(t)\}_{i=1}^n$ for $\{Q_1, \cdots, Q_n\}$. The only difference from before is that $P_{i,0} = Q_0$ and $P_{n,2} = Q_{n+1}$. Figure 4 shows an example of such a curve when $n = 4$.

7. Results

Our curves extend the geometric properties of [1] by introducing rational weight parameters and improve the optimization procedure for these curves. Figure 3 shows the effect of choosing different rational weights on the shape of the curve. In each case the weights for all control points are uniform though the user could set weights on a control point by control point basis. As the weights become higher, the curves tend to become sharper, though still smooth. When all weights are one, we produce the piece-wise polynomial κ-curves [1]. The right portion of figure 3 shows the result of our minimum eccentricity weights with the bottom right figure reproducing a circle as the four points are co-circular even though they are non-uniformly spaced on a circle.

Our minimum eccentricity weights do not require that the user set any weights individually on the curve. Instead, our method chooses the weights automatically through our optimization. At the same time, the minimum eccentricity weights become unstable as weights approach zero, which leads to our clamped solution. Figure 5 shows the results of clamping our minimum eccentricity weights compared to κ-curves [1]. In both cases, our minimum eccentricity weights provide a "fairer" curve as measured by variation of curvature. At the same time, all curves have the property that the local maximum of curvature is reached at the control points even though the (absolute value of) curvature is near constant for the far left image. Each curve is also curvature continuous everywhere except at inflection points.

In figure 5 (middle), our clamped minimum eccentricity weights produce a solution similar to κ-curves though a bit more round. In regions where curves

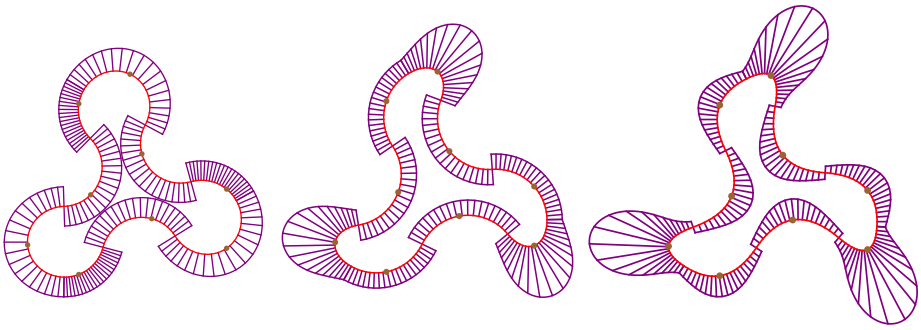

Figure 5: The purple lines show curvature normals. Each picture uses the same input points with different weight functions. From left to right: min-eccentricity weights, clamped min-eccentricity weights, and the original κ-curve.

have concave corners, our clamped minimum eccentricity weights often produce shapes that locally resemble κ-curves. However, figure 6 shows an example with mostly convex (though overlapping) curves. In these cases, our clamped minimum eccentricity weights appears far closer to the unclamped version of the curves than κ-curves. Figure 7 shows the results of our method on a non-symmetric shape with non-uniform control point spacing from [7]. Figure 8 demonstrates our minimum eccentricity weights where the user modifies the parameter μ_i at a single control point to make the curve appear sharper. Figure 9 shows our method creating a cusp as points move closer to one another. Cusps represent local maxima of curvature and, hence, can only appear at input points.

Like κ-curves, our solution is a global solution, which means the whole curve will change after a small movement of a control point. However, in practice, the influence of a control point drops dramatically away from that point. Figure 10 shows an example of this effect using our minimum eccentricity weights. In both cases the original curve is shown in blue with the newly modified red curve drawn on top. The resulting curve only changes within a small region away from the initial control point in any significant fashion.

Our paper also improves upon the original κ-curve optimization by constructing a formal energy that can be minimized in addition to extending the method to rational curves. Figure 11 shows the results of this optimization with the top row having a uniform weight of one and the bottom using our minimum eccentricity weights. The starting curve appears on the left and its error is listed below. After performing 60 iterations of our initial guess from Section 5.1, we obtain the curve in the middle. For many applications, this curve may be a sufficiently good approximation. However, optimizing our energy function produces a significant reduction in error as shown on the right, which does affect the shape of the curve and is particularly visible on the bottom row.

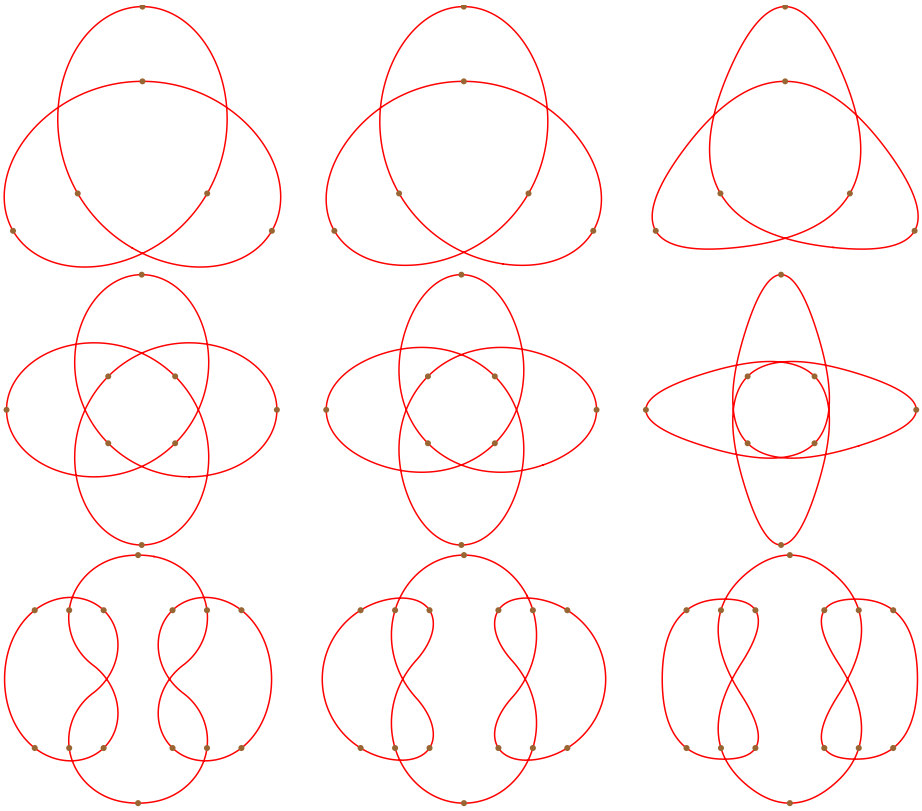

Figure 6: Comparisons between different weight choices. Left to right: min eccentricity weights, clamped min eccentricity weights, original κ-curve.

8. Limitations

In our construction we require that the rational weights w_i for all off-curve Bézier control points $P_{i,1}$ be positive. The implication of this requirements is that all pieces of the rational quadratic curve are minor arcs. Hence, in the elliptical case, the arc cannot span more than half of an ellipse. This restriction implies that when all input points are on a circle, the curve produced by our minimum eccentricity method may not form a circle in some scenarios. In particular, when the end-points of each Bézier curve $P_{i,0}, P_{i,2}$ span more than half a circle, we cannot produce an exact circle.

The connection between the end-points of each Bézier curve and the control points the user specifies is, unfortunately, governed by a non-linear optimization. Let θ be the angle between consecutive input points formed from the center of the circumscribed circle as shown in figure 12. If the angle spanned by two consecutive segments is less than π, for all segments, then the end-points of each Bézier curve cannot span more than half a circle since $P_{i,0}$ lies in between

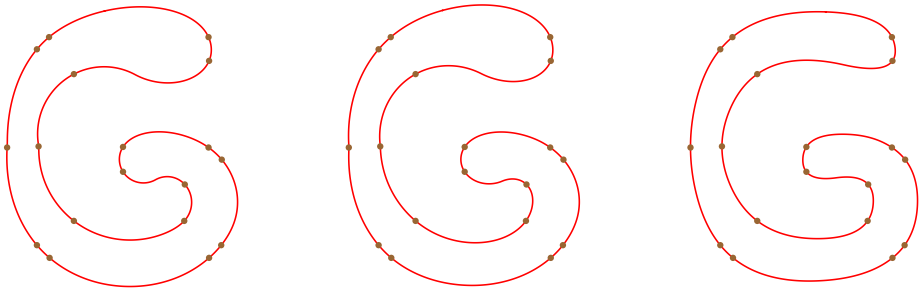

Figure 7: An asymmetric example with non-uniformly spaced control points. From left to right: min eccentricity weights, clamped min eccentricity weights, original κ-curve.

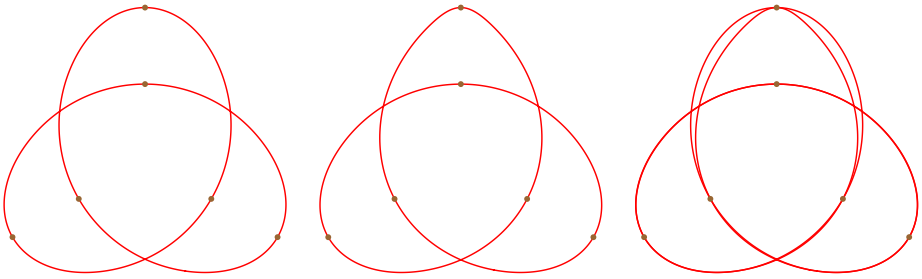

Figure 8: Clamped minimum eccentricity weight curves with tension values μ_i set to 1 (left) and with the weight of the top input point set to 2 (middle). The right picture shows the overlapping curves for comparison.

Q_{i-1} and Q_i. Yet this property is too restrictive in practice. We tested our technique on many different point distributions of input points Q_i on a circle. In our tests we found that if the angle spanned by consecutive input points was less than π, our method reproduced the circumscribed circle. However, our method would often fail to produce a circle if the angle spanned by consecutive input points exceeded π. Figure 12 shows several examples where non-uniform point distributions produce circles and two failure cases where our method was unable to reproduce a circle.

Acknowledgements

This work was supported in part by a gift from Adobe Systems Incorporated.

[1] Z. Yan, S. Schiller, G. Wilensky, N. Carr, S. Schaefer, k-curves: interpolation at local maximum curvature, ACM Transactions on Graphics (TOG) 36 (4) (2017) 129.

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

Figure 9: The creation of a cusp using clamped min eccentricity weights. Moving the 2^{nd} and 4^{th} points closer creates a cusp at an input point.

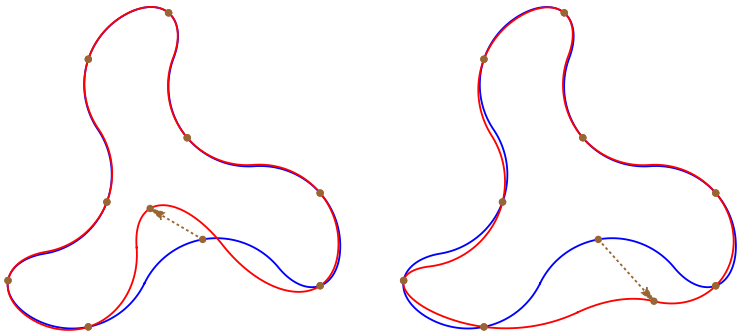

Figure 10: The blue/red curves show our curves before/after moving a single input point. Despite the global nature of the optimization, the change in the curve tends to decrease with distance from the input point.

[3] G. Farin, N. Sapidis, Curvature and the fairness of curves and surfaces, IEEE Computer Graphics and Applications 9 (2) (1989) 52–57.

[4] S. Havemann, J. Edelsbrunner, P. Wagner, D. Fellner, Curvature-controlled curve editing using piecewise clothoid curves, Computers & Graphics 37 (6) (2013) 764 – 773.

[5] G. Farin, Curves and Surfaces for CAGD: A Practical Guide, 5th Edition, Morgan Kaufmann Publishers Inc., 2002.

[6] E. Catmull, R. Rom, A class of local interpolating splines, in: Computer aided geometric design, Elsevier, 1974, pp. 317–326.

[7] N. Dyn, D. Levin, J. A. Gregory, A 4-point interpolatory subdivision scheme for curve design, Computer aided geometric design 4 (4) (1987) 257–268.

[8] R. Schaback, Interpolation with piecewise quadratic visually C^2 Bézier polynomials, Computer Aided Geometric Design 6 (3) (1989) 219–233.

[9] Y. Y. Feng, J. Kozak, On G^2 continuous interpolatory composite quadratic Bézier curves, Journal of Computational and Applied Mathematics 72 (1) (1996) 141–159.

[10] J. Hoschek, Circular splines, Computer-Aided Design 24 (11) (1992) 611–618.

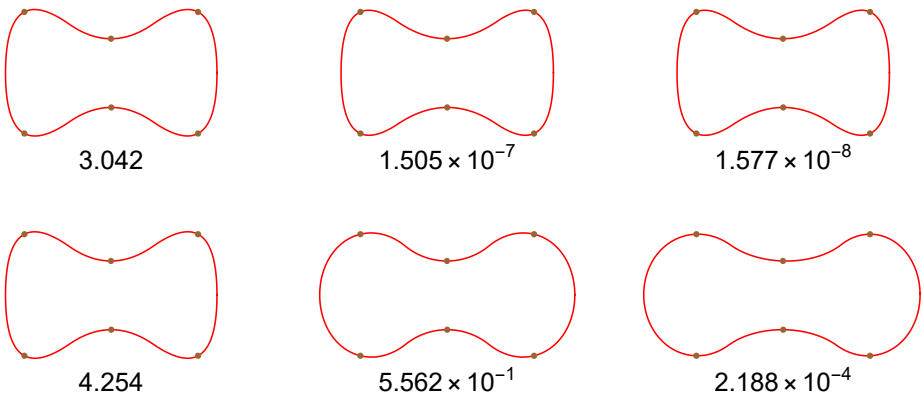

Figure 11: Curve optimization. Each image has the same input points with the error shown below. Each row shows the initial guess (left), after 60 iterations of refining the initial guess (middle), and the result of our optimization (right). The first row uses weight 1 while the second row are our clamped min eccentricity weights. The top middle picture corresponds to the κ-curve result in [1].

[11] D. S. Meek, D. J. Walton, Approximation of discrete data by g1 arc splines, Computer-Aided Design 24 (6) (1992) 301–306.

[12] M. K. Yeung, D. J. Walton, Curve fitting with arc splines for nc toolpath generation, Computer-Aided Design 26 (11) (1994) 845–849.

[13] A. Kurnosenko, Biarcs and bilens, Computer Aided Geometric Design 30 (3) (2013) 310–330.

[14] D. Meek, D. Walton, Planar G^2 Hermite interpolation with some fair, c-shaped curves, Journal of Computational and Applied Mathematics 139 (1) (2002) 141–161.

[15] L. A. Piegl, W. Tiller, Data approximation using biarcs, Engineering with computers 18 (1) (2002) 59–65.

[16] H.-J. Wenz, Interpolation of curve data by blended generalized circles, Computer Aided Geometric Design 13 (8) (1996) 673–680.

[17] C. H. Séquin, K. Lee, J. Yen, Fair, G^2- and C^2- continuous circle splines for the interpolation of sparse data points, Computer-Aided Design 37 (2) (2005) 201–211.

[18] C. Sun, H. Zhao, Generating fair, C^2 continuous splines by blending conics, Computers & Graphics 33 (2) (2009) 173–180.

[19] S. Schaefer, A factored interpolatory subdivision scheme for surfaces of revolution, Master's thesis, Rice University (1993).

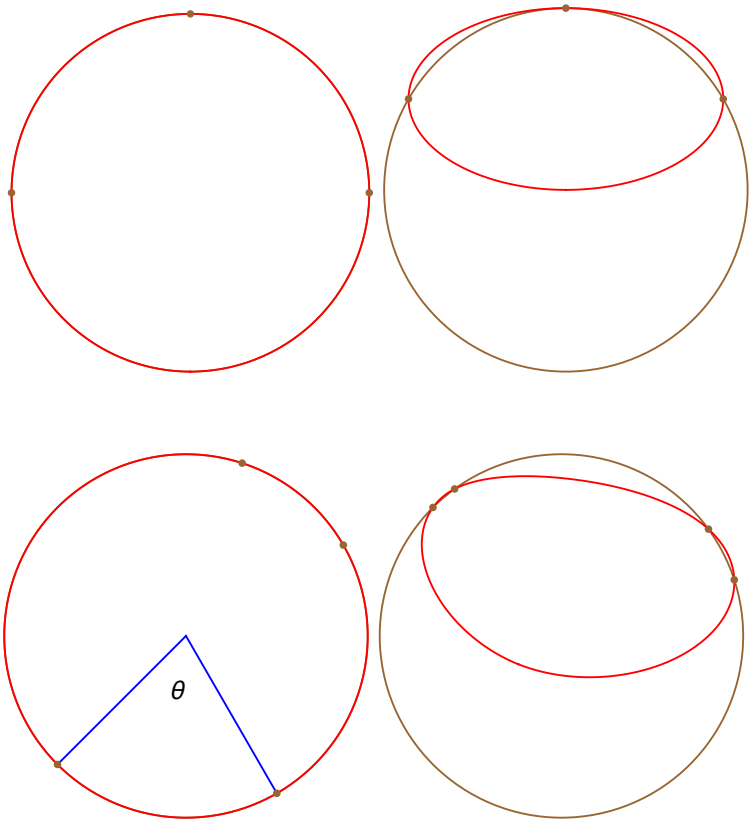

Figure 12: Circle reproduction with three (top) or four points (bottom). The left column has input points with central angles less than π. The right has points with one central angle over π, which leads to a non-circular curve.

[20] C. Xu, T.-w. Kim, G. Farin, The eccentricity of conic sections formulated as rational Bézier quadratics, Computer Aided Geometric Design 27 (6) (2010) 458–460.

[21] A. Cantón, L. Fernández-Jambrina, E. R. María, Geometric characteristics of conics in Bézier form, Computer-Aided Design 43 (11) (2011) 1413–1421.

[22] R. Schaback, Planar curve interpolation by piecewise conics of arbitrary type, Constructive Approximation 9 (4) (1993) 373–389.

[23] X. Yang, Curve fitting and fairing using conic splines, Computer-Aided Design 36 (5) (2004) 461–472.

[24] Y. Ahn, H. Kim, Curvatures of the quadratic rational Bézier curves, Computers & Mathematics with Applications 36 (9) (1998) 71–83.

[25] A. G. Akritas, Sylvester's form of the resultant and the matrix-triangularization subresultant prs method, in: Computer Aided Proofs in Analysis, Springer, 1991, pp. 5–11.

[26] A. B. Ayoub, The eccentricity of a conic section, The College Mathematics Journal 34 (2) (2003) 116.

[27] J. M. Lane, R. F. Riesenfeld, Bounds on a polynomial, BIT Numerical Mathematics 21 (1) (1981) 112–117.

Appendix A. Uniqueness of the root of equation (7)

Proof. The right-hand-side (RHS) of equation 7 is a polynomial of degree 4. If we use the notation that $v_0 = P_0 - Q$ and $v_2 = P_2 - Q$, the Bézier coefficients of this polynomial are

$$(-w_0^2 w_1 |v_0|^2, -\frac{1}{4} w_0^2 w_2 v_0 \cdot (v_0 + v_2), 0, \frac{1}{4} w_0 w_2^2 v_2 \cdot (v_0 + v_2), w_1 w_2^2 |v_2|^2) \quad \text{(A.1)}$$

where $v_0 \neq 0, v_2 \neq 0$, and $w_i > 0$ for $i = 0, 1, 2$.

We want to show that there is a unique root t in $[0, 1]$. When $t = 0$, the value of the RHS of equation7 is the first coefficient of equation A.1, which is negative. When $t = 1$, the value is the last coefficient of equation A.1, which is positive. So the there must be at least one root t between zero and one because the function is continuous.

To prove the uniqueness, we use the fact that polynomials in Bézier form follow Descartes' rule of signs for bounding the number of real roots of a polynomial [27], which means the number of real roots is less than or equal to the number of sign changes in the sequence of the polynomial's coefficients. In our problem, we have five coefficients with the first being negative, the middle 0, and the last positive. Since weights w_i are always positive, we only need to know the sign of $v_0 \cdot (v_0 + v_2)$ and $v_2 \cdot (v_0 + v_2)$. If one of these coefficients is zero, then no matter what the other is, then the sign changes in the coefficients is one. Hence, there would be at most one root in $[0, 1]$. If both coefficients are non-zero, then the number of sign changes of the coefficients are always one unless both $v_0 \cdot (v_0 + v_2)$ and $v_2 \cdot (v_0 + v_2)$ are negative, which is not possible because the sum is non-negative:

$$v_0 \cdot (v_0 + v_2) + v_2 \cdot (v_0 + v_2) = |v_0 + v_2|^2 \geq 0 \quad \text{(A.2)}$$

Therefore, the number of times the coefficients change signs in equation A.1 is always one, and there is exactly one root in $[0, 1]$ for equation 7. □