Fast approaches to simplify and offset Bézier curves within specified error limits

Fabian Yzerman

https://blend2d.com/

Based on a master's thesis - Revision 1.1

January 18, 2020

Abstract

Several algorithms to flatten and offset planar Bézier curves will be developed. These include quadratic, cubic and rational quadratic curves. Because of the fast approaches for quadratic Bézier curves, cubic and rational quadratic curves are approximated by ordinary quadratic ones for further processing. An estimation of the error bound is given by the maximum euclidean distance. Finally, the performance characteristics of the methods will be compared to existing ones.

Contents

Chapter 1

Introduction

A Bézier curve is a type of freeform curve that is used to precisely define smooth paths by interpolating between a number of given control points. These curves are found in a wide array of different applications in computer graphics and usually require additional processing such as flattening and offsetting to be drawn on the screen.
The term flattening refers to the approximation of a curve with line segments or a polyline so that it can be used in traditional scanline rendering techniques. Offsetting is needed when computing offset curves for the drawing of uniformly thick curves, which is commonly known as stroking. In figure 1.1 we see a cubic Bézier curve (green) with the control polygon (gray), its polyline approximation (red) and the resulting offset polyline approximation to both sides of the curve (blue).

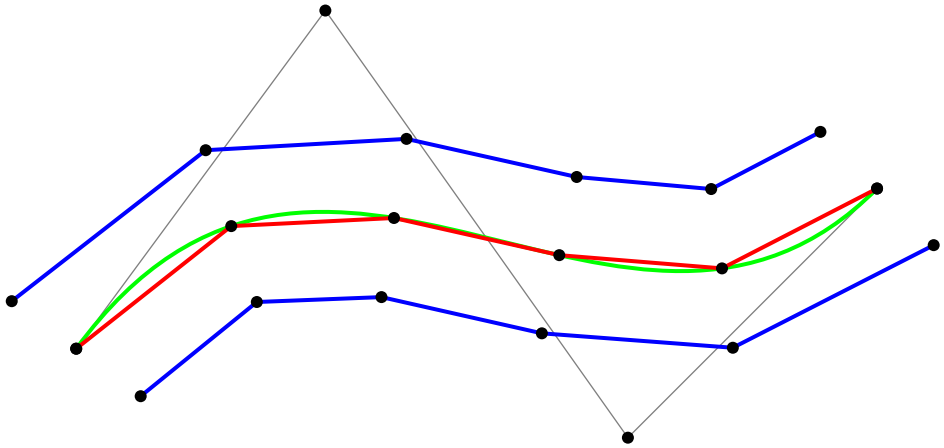

Figure 1.1: Flattening and offsetting of a cubic curve

In this thesis we will cover the most common planar Bézier curves. These are quadratic, cubic and rational quadratic curves. Many different methods to flatten and offset those already exist, some of which will be mentioned in the following.

1.1 Related work

The simplest forms of flattening are incremental approaches with uniformly spaced parameter intervals. They involve direct evaluation of the points or alternatively forward differencing. These techniques are very fast but generally do not provide any error bounds. An adaptive forward differencing method that is adjusting the step size to approximately one pixel has been developed [1].
Another pragmatic approach is recursive subdivision. The different implementations are closely related to De Casteljau's algorithm and mostly differ in their termination condition [2][3][4][5]. Although recursive subdivisioning is able to provide error bounds it usually suffers from overapproximation and therefore unnecessary line segments.
Hain et al. described iterative approaches which attempt to minimize the total number of approximated line segments. These include the flattening of cubic Bézier curves [6][7] and their offset curves [8].
Degree reduction techniques may also be used for the purpose of flattening Bézier curves [4]. Early implementations focused on minimizing the maximum euclidean distance between two curves [9]. However, it was suggested that least squares methods are generally faster and give better results [10]. More recent approaches offer better multi-degree reduction and continuity control [11].
An extensive comparison of different offset curve approximation methods has been done by Elber et al. [12]. Furthermore, we discussed the offsetting of quadratic Bézier curves in a recent paper [13].

1.2 Motivation

Basic benchmarking suggests that, compared to the whole rendering process, the time spent on flattening Bézier curves or their offsets is significant. Therefore, improving the total throughput would certainly benefit the overall drawing performance.
The general idea is to focus on incremental approaches that also provide error bounds by using the maximum euclidean distance

$$\max_{t\in[0,1]} \|C_b(t) - C_a(t)\|$$

between two Bézier curves $C_a(t)$ and $C_b(t)$ as the approximation error. We are going to find constant parameter values for t and utilize the maximum distance to flatten quadratic curves with a fixed parameter step. Unlike e.g. cubic Bézier curves (see figure 1.1), quadratic curves retain the same maximum distance over an uniformly distributed parameter interval. This effectively means that every point of a flattened

quadratic curve is instantly given. A similar observation can also be made for the distance between cubic curves and quadratic curves.
The goal of this thesis is to investigate all types of curves that are necessary to define paths for SVG (Scalable Vector Graphics) [14]. These obviously include quadratic and cubic Bézier curves as described in the reference. Furthermore, elliptical arcs may be affinely transformed from one or multiple unit circle segments which are precisely defined by rational quadratic Bézier curves.
We will introduce fast approaches to flatten and offset quadratic Bézier curves. Cubic and rational quadratic curves are going to be approximated by ordinary quadratic curves so that the fast methods can be applied afterwards. Additionally, this process might also be used for other applications such as converting from OpenType glyphs to TrueType glyphs, i.e. approximating cubic Bézier curves with quadratic ones.
As to stroking, offset curves do not generally represent the proper outline that is defined by SVG. However, the standard explicitly mentions that strokes of a tight curve may be different across platforms. This is referring to the hole on the inner path that appears when filling the area between offset curves (see left shape of figure 1.2).

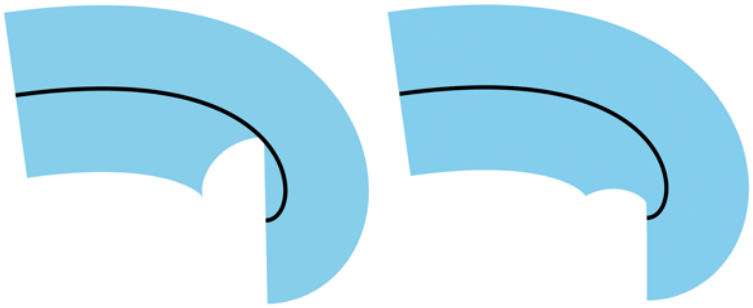

Figure 1.2: Difference of stroke implementations (from the SVG Reference)

In addition to simple and fast implementations, the incremental approaches also have another interesting property that might benefit the animating capabilities of SVG. Small changes of a curve's control points will further result in small changes of the approximated output, similar to Hain's iterative techniques. With subdivision new line segments suddenly pop in when the quality criterion is not met for the current segment of the curve. In the case of relatively high error bounds and animations these pop-ins could be noticed by the observer.

Chapter 2

Preliminaries

This chapter serves as a compilation of definitions and terminology about Bézier curves which differentiate into polynomial and rational curves. Bézier curves are parametric curves that are commonly used in computer graphics and geometric modeling. They are named after Pierre Bézier who intended to use them for automobile design at the french carmaker Renault.

2.1 Bernstein basis polynomials

The Bernstein basis polynomials $b_{i,n}$ of degree n are defined as

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

with the binomial coefficient $\binom{n}{i}$ and $0 \leq i \leq n$. Restricted to the parameter $t \in [0,1]$, polynomials in Bernstein form are especially important for the definition of Bézier curves. A thorough retrospective on the matter has been composed by Farouki [15].

2.2 Bézier curves

A planar (polynomial) Bézier curve $C_n(t)$ of degree n is defined by

$$C_n(t) = \sum_{i=0}^{n} b_{i,n}(t) P_i$$

with the parameter $t \in [0,1]$ and $n+1$ control points $P_i = (x_i, y_i)$. It interpolates the resulting curve between the given points and runs from P_0 ($t = 0$) to P_n ($t = 1$). So for $n = 1$ the expression yields a linear interpolation between the two endpoints:

$$C_1(t) = (1-t)P_0 + tP_1$$

The quadratic ($n = 2$) and cubic ($n = 3$) forms are given by:

$$C_2(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$
$$C_3(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

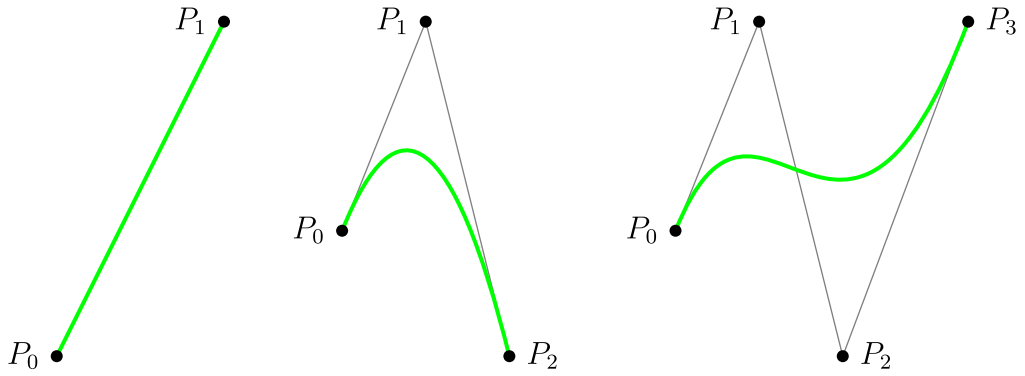

Figure 2.1: Linear, quadratic and cubic Bézier curves

We can also write Bézier curves in polynomial canonical form

$$C_1(t) = (P_1 - P_0)t + P_0$$
$$C_2(t) = (P_2 - 2P_1 + P_0)t^2 + (2P_1 - 2P_0)t + P_0$$
$$C_3(t) = (P_3 - 3P_2 + 3P_1 - P_0)t^3 + (3P_2 - 6P_1 + 3P_0)t^2 + (3P_1 - 3P_0)t + P_0$$

or alternatively with coefficients $Q_{i,n} = (x_i, y_i)$:

$$\begin{aligned} C_n(t) &= \sum_{i=0}^{n} Q_{i,n} t^{n-i} \\ &= Q_{0,n}t^n + Q_{1,n}t^{n-1} + \cdots + Q_{n,n} \end{aligned} \tag{2.1}$$

From now on we will simply refer to Bézier curves as curves.

2.3 Rational Bézier curves

A planar rational Bézier curve $C_{r,n}(t)$ of degree n is defined by

$$C_{r,n}(t) = \frac{\sum_{i=0}^{n} b_{i,n}(t) w_i P_i}{\sum_{i=0}^{n} b_{i,n}(t) w_i}$$

with the parameter $t \in [0, 1]$, $n + 1$ control points $P_i = (x_i, y_i)$ and weights $w_i > 0$. Due to the addition of weights, rational curves are able to represent conic sections. Detailed works on their geometric properties have been published [16][17][18].

The quadratic form ($n = 2$) is given by:

$$C_{r,2}(t) = \frac{(1-t)^2 w_0 P_0 + 2t(1-t)w_1 P_1 + t^2 w_2 P_2}{(1-t)^2 w_0 + 2t(1-t)w_1 + t^2 w_2}$$

By reparameterizing [19] the curve we obtain the quadratic standard form

$$C_r(t) = \frac{(1-t)^2 P_0 + 2t(1-t)wP_1 + t^2 P_2}{(1-t)^2 + 2t(1-t)w + t^2}$$

with:

$$w = \frac{w_1}{\sqrt{w_0 w_2}}$$

It should be noted that reparameterized rational curves might look the same but are not equivalent in terms of evaluated points depending on the parameter t. However, we are now able to differentiate the type of conic section between elliptic ($w < 1$), parabolic ($w = 1$) and hyperbolic ($w > 1$).

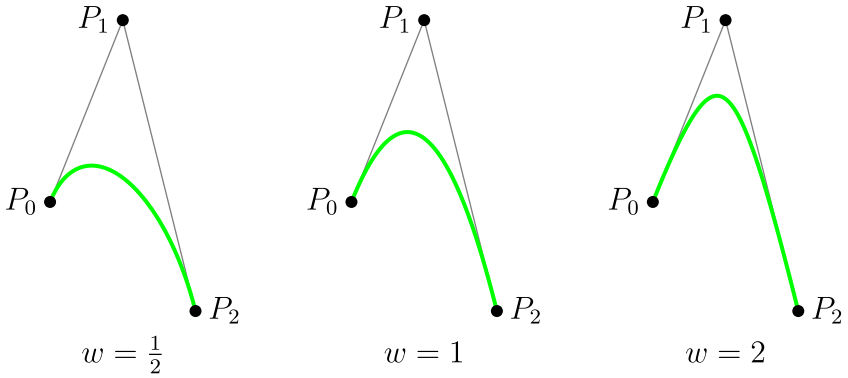

Figure 2.2: Rational quadratic Bézier curves with different weights

Planar rational curves with the points $P_i = (x_i, y_i)$ may be expressed as polynomial space curves with homogeneous coordinates P_i^* (perspective projection) such that:

$$P_i^* = w_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} x_i^* \\ y_i^* \\ w_i \end{pmatrix} \iff P_i = \frac{1}{w_i} \begin{pmatrix} x_i^* \\ y_i^* \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{2.2}$$

Thus, a polynomial space curve defines:

$$\begin{pmatrix} x(t) \\ y(t) \\ w(t) \end{pmatrix} = \sum_{i=0}^{n} b_{i,n}(t) P_i^* = \sum_{i=0}^{n} b_{i,n}(t) w_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{2.3}$$

We apply the transformation of (2.2) to (2.3) yielding

$$\frac{1}{w(t)}\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \frac{\sum_{i=0}^{n} b_{i,n}(t) w_i P_i}{\sum_{i=0}^{n} b_{i,n}(t) w_i}$$

to reveal that the terms are indeed equivalent.
In the course of this thesis there is no need for rational curves of a higher degree than quadratic. So for the sake of simplicity, we will now refer to rational quadratic Bézier curves as rational curves.

2.4 De Casteljau's algorithm

The most common way to evaluate curves at specific parameter values is by using De Casteljau's algorithm. It is named after Paul de Casteljau, who in particular studied the mathematical nature of the curves. The method works by recursively interpolating between the points of a curve [4].
De Casteljau's algorithm is defined by the recursion rule

$$P_i^{(k)}(t) = (1-t)P_i^{(k-1)}(t) + tP_{i+1}^{(k-1)}(t) \quad \begin{cases} k = 1, \ldots, n \\ i = 0, \ldots, n-k \end{cases} \tag{2.4}$$

with $P_i^{(0)}(t) = P_i$ as the starting point and $P_i = (x_i, y_i)$ for polynomial curves or $P_i = (x_i^*, y_i^*, w_i)$ for rational curves (see previous section). Then $P_0^{(n)}(t)$ is the evaluated point on the curve. Additionally, this term is equivalent to polynomial representations of the curve, i.e. $P_0^{(n)}(t) = C_n(t)$.

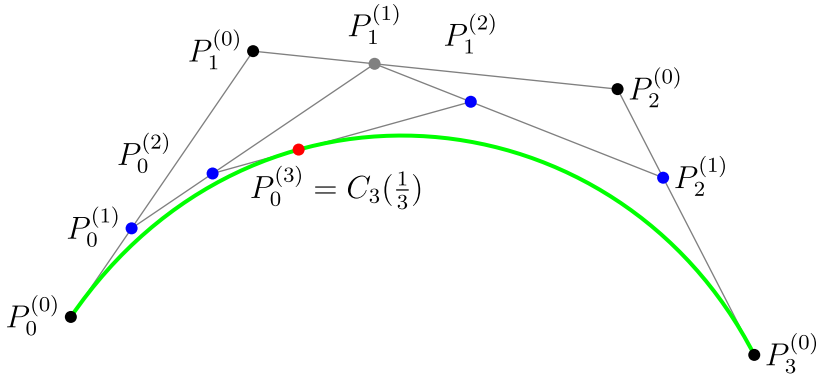

Figure 2.3: De Casteljau's algorithm applied to a cubic curve at $t = \frac{1}{3}$

For example

$$
\begin{array}{llll}
P_0^{(0)} & & & \\
P_1^{(0)} & P_0^{(1)} & & \\
P_2^{(0)} & P_1^{(1)} & P_0^{(2)} & \\
P_3^{(0)} & P_2^{(1)} & P_1^{(2)} & P_0^{(3)}
\end{array}
$$

are the unrolled recursion steps of a cubic curve. Figure 2.3 visualizes the interpolated points. In addition to the evaluated point on the curve (red), we also obtain the control points (blue) of the curve segments $[0, t]$ (before) and $[t, 1]$ (after). Thus, in the current case, we are effectively subdividing the curve into two subcurves with the points $P_0^{(0)}, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}$ and $P_0^{(3)}, P_1^{(2)}, P_2^{(1)}, P_3^{(0)}$.

2.5 Blossoming

In the following chapter we will need to subdivide a curve between the two arbitrary parameters $\{t_a, t_b\} \in [0, 1]$. With the use of so called *blossoming* [4][15] it is possible to get the curve segment within the parameter interval $[t_a, t_b]$. Instead of interpolating the points between the same parameter t, a new parameter t_k is used for each step k of the recursion. So adapting (2.4) gives:

$$
P_i^{(k)}(t_1, \ldots, t_k) = (1 - t_k) P_i^{(k-1)}(t_{k-1}) + t_k P_{i+1}^{(k-1)}(t_{k-1}) \quad \begin{cases} k = 1, \ldots, n \\ i = 0, \ldots, n - k \end{cases}
$$

Again the unrolled recursion steps of a cubic curve are:

$$
\begin{array}{llll}
P_0^{(0)} & & & \\
P_1^{(0)} & P_0^{(1)}(t_1) & & \\
P_2^{(0)} & P_1^{(1)}(t_1) & P_0^{(2)}(t_1, t_2) & \\
P_3^{(0)} & P_2^{(1)}(t_1) & P_1^{(2)}(t_1, t_2) & P_0^{(3)}(t_1, t_2, t_3)
\end{array}
$$

Then the points $\breve{P}_i$ of the subdivided curve of degree n are defined by

$$
\breve{P}_i = P_i^{(n)}(\underbrace{t_a, t_a, \ldots}_{n-i}, \underbrace{t_b, t_b, \ldots}_{i}) \tag{2.5}
$$

with $0 \leq i \leq n$ and $t_a \leq t_b$.
Thus, the middle segment of e.g. a cubic curve would have the points $P_0^{(3)}(t_a, t_a, t_a)$, $P_0^{(3)}(t_a, t_a, t_b)$, $P_0^{(3)}(t_a, t_b, t_b)$ and $P_0^{(3)}(t_b, t_b, t_b)$.

2.6 Parametric distance

We define parametric distance as the euclidean distance between two curves $C_a(t)$ and $C_b(t)$, both using the same parameter t to keep the expression simple:

$$d_{a,b}(t) = \|C_b(t) - C_a(t)\| \tag{2.6}$$

Now let $C_a(t)$ be approximated by $C_b(t)$. Then

$$\max_{t\in[0,1]} d_{a,b}(t) \tag{2.7}$$

is the error bound of the approximation. With the directional Hausdorff distance

$$\check{H}(A,B) = \max_{a\in A}\min_{b\in B}\|b-a\|$$

and d_H as the Hausdorff distance between the two curves we easily show that:

$$\begin{aligned} d_H &= \max_{t_a\in[0,1]}\min_{t_b\in[0,1]}\|C_b(t_b) - C_a(t_a)\| \\ &\leq \max_{t\in[0,1]}\|C_b(t) - C_a(t)\| \end{aligned}$$

The inequality

$$d_H \leq \max_{t\in[0,1]} d_{a,b}(t) \tag{2.8}$$

means that (2.7) is an upper bound of d_H.

Chapter 3

Simplifying

In the following chapter we will utilize parametric distance to approximate quadratic, cubic and rational curves by simpler curve segments. The general approach is to study the maximum error bound of a single curve first and then deduce a parameter interval t_Δ in which the curve may be subdivided and approximated within a given error limit. Since the term degree reduction does not apply for the rational case, we will call this process *simplifying* a curve.

3.1 Maximum error of quadratic curves

Let $C_1(t)$ be a linear approximation of a quadratic curve $C_2(t)$ which has P_0 and P_2 as their mutual points:

$$C_2(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$
$$C_1(t) = (1-t)P_0 + tP_2$$

We want to find the maximum of the parametric distance (2.6) to get the error bound:

$$d_{max,2} = \max_{t\in[0,1]} d_{1,2}(t)$$

Symbolically solving $d'_{1,2}(t) = 0$ with Mathematica yields $t_{max} = \frac{1}{2}$. This parameter has to be a maximum because $d(t) = 0$ for $t \in \{0, 1\}$ and $d(t) \geq 0$ for $t \in [0, 1]$. Therefore, the maximum error of a single quadratic curve is

$$d_{max,2} = \tfrac{1}{4}\|P_0 - 2P_1 + P_2\| \tag{3.1}$$

which is an upper bound according to (2.8) and does not depend on the parameter t. The last part of the expression is equivalent to the first coefficient of the polynomial

canonical form (2.1). So we are able to simplify it further:

$$d_{max,2} = \tfrac{1}{4}\|Q_{0,2}\|$$

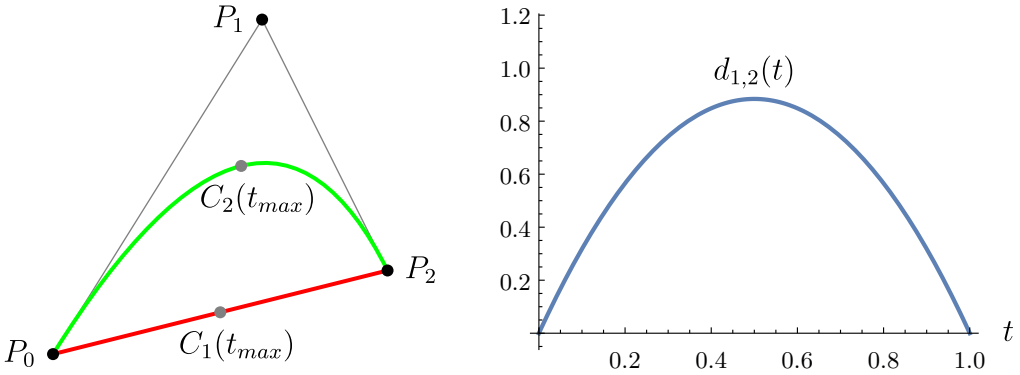

Figure 3.1: Parametric distance of a quadratic curve

3.2 Subdivision of quadratic curves

Instead of measuring the maximum error for the quadratic curve, we want to limit a specific error by subdividing the curve into multiple segments. Let $\breve{C}_2(t)$ be a curve that is arbitrarily subdivided between the parameters t_a and t_b.
With (2.5) we are able to express the points of the subcurve as follows:

$$\breve{P}_0 = P_0^{(2)}(t_a, t_a)$$
$$\breve{P}_1 = P_0^{(2)}(t_a, t_b)$$
$$\breve{P}_2 = P_0^{(2)}(t_b, t_b)$$

Now let $t_b = t_a + t_\Delta$ such that we have a curve segment of constant parameter step t_Δ and insert $\breve{P}_0, \breve{P}_1$ and $\breve{P}_2$ into (3.1). Interestingly, t_a and t_b vanish once the expression is simplified so that

$$\begin{aligned} d_{max,2} &= \tfrac{1}{4}\|\breve{P}_0 - 2\breve{P}_1 + \breve{P}_2\| \\ &= \tfrac{1}{4}\|P_0 - 2P_1 + P_2\|t_\Delta^2 \end{aligned}$$

which reveals the original points P_0, P_1 and P_2 of the curve $C_2(t)$.
Then solving for t_Δ yields:

$$t_\Delta = \sqrt{\frac{4\, d_{max,2}}{\|P_0 - 2P_1 + P_2\|}} = \sqrt{\frac{4\, d_{max,2}}{\|Q_{0,2}\|}}$$

This means that for any parameter $t \in [0,1]$ on the quadratic curve $C_2(t)$ we are able to draw a linear curve segment to $C_2(t + t_\Delta)$ which retains the maximum parametric distance $d_{max,2}$ to the quadratic segment.
Remark. Reducing a quadratic curve to linear curve segments, or rather line segments, is basically the same as flattening. We will come back to this topic in the next chapter.

3.3 Maximum error of cubic curves

Let $C_2(t)$ be a quadratic approximation of a cubic curve $C_3(t)$. Similar to the previous part, we require that the curves share the same endpoints P_0 and P_3:

$$C_3(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$
$$C_2(t) = (1-t)^2 P_0 + 2t(1-t)P_c + t^2 P_3$$

However, the point P_c is still unknown and needs to be parameterized. By elevating the degree [4] of a quadratic curve we are able to find such a parameterization.
The elevated points $\hat{P}_i$ are defined as

$$\hat{P}_i = \left(1 - \frac{i}{n+1}\right) P_i + \left(\frac{i}{n+1}\right) P_{i-1}$$

with $1 \leq i \leq n$. So for a quadratic curve it gives:

$$\hat{P}_0 = P_0 \tag{3.2}$$
$$\hat{P}_1 = \tfrac{2}{3} P_1 + \tfrac{1}{3} P_0 \tag{3.3}$$
$$\hat{P}_2 = \tfrac{1}{3} P_2 + \tfrac{2}{3} P_1 \tag{3.4}$$
$$\hat{P}_3 = P_2 \tag{3.5}$$

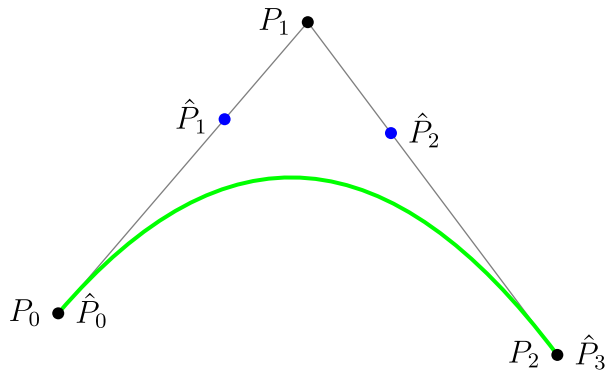

Figure 3.2: Degree elevation of a quadratic curve

After inserting (3.2) into (3.3) and (3.5) into (3.4) we solve both equations for P_1:

$$\hat{P}_1 = \tfrac{2}{3}P_1 + \tfrac{1}{3}\hat{P}_0 \iff P_1 = \tfrac{3}{2}\hat{P}_1 - \tfrac{1}{2}\hat{P}_0 = \left(1 - \tfrac{3}{2}\right)\hat{P}_0 + \tfrac{3}{2}\hat{P}_1$$
$$\hat{P}_2 = \tfrac{1}{3}\hat{P}_3 + \tfrac{2}{3}P_1 \iff P_1 = \tfrac{3}{2}\hat{P}_2 - \tfrac{1}{2}\hat{P}_3 = \left(1 - \tfrac{3}{2}\right)\hat{P}_3 + \tfrac{3}{2}\hat{P}_2$$

Then replacing the elevated points with the points of $C_3(t)$ yields the candidates:

$$P_{c,1} = \left(1 - \tfrac{3}{2}\right)P_0 + \tfrac{3}{2}P_1$$
$$P_{c,2} = \left(1 - \tfrac{3}{2}\right)P_3 + \tfrac{3}{2}P_2$$

These parameterizations ensure first-order continuity for non-degenerate curves either at the start or at the end of $C_3(t)$. Like in the quadratic case, we symbolically solve the derivative of the parametric distance for t and define $d_{c,1}(t)$ and $d_{c,2}(t)$ with their respective candidates. Thus, the first parameterization gives $t_{max} = \frac{2}{3}$ and the second one $t_{max} = \frac{1}{3}$. We notice that both parameterizations have the same error bound

$$\begin{aligned} d_{max,3} &= \tfrac{4}{27}\|P_0 - 3P_1 + 3P_2 - P_3\| \\ &= \tfrac{4}{27}\|Q_{0,3}\| \end{aligned}$$

and, just as the quadratic case, the dependency of the first coefficient of the curve.

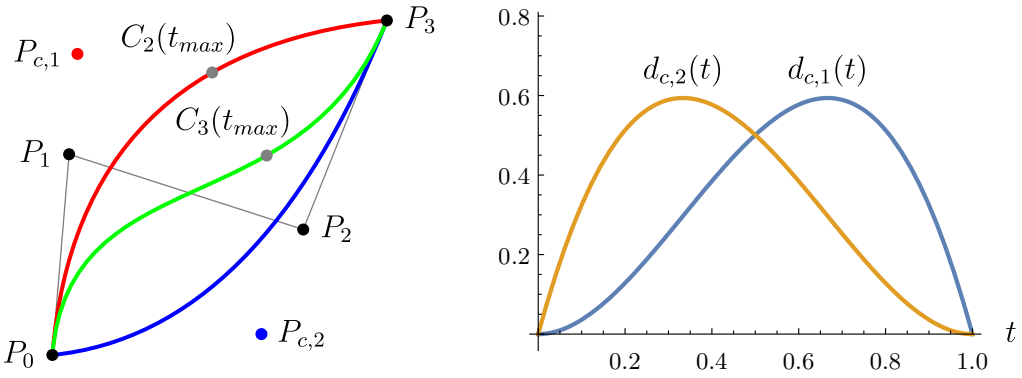

Figure 3.3: Parametric distance of a cubic curve (I)

However, if we give up on continuity, we are able to get a better approximation. By taking the center point of the two previous control points such that

$$\begin{aligned} P_{c,3} &= \left(1 - \tfrac{1}{2}\right)P_{c,1} + \tfrac{1}{2}P_{c,2} \\ &= \tfrac{1}{4}(3P_1 - P_0 + 3P_2 - P_3) \end{aligned}$$

we get a so called *midpoint approximation* of C_3 because $C_3(\frac{1}{2}) = C_2(\frac{1}{2})$. Again, we define $d_{c,3}(t)$ and symbolically solve $d'_{c,3}(t) = 0$ for t so that we get

$$t_{max} = \tfrac{1}{6}(3 \pm \sqrt{3})$$

and the corresponding maximum parametric distance:

$$\begin{aligned} d_{max,3} &= \tfrac{1}{12\sqrt{3}}\|P_0 - 3P_1 + 3P_2 - P_3\| \\ &= \tfrac{1}{12\sqrt{3}}\|Q_{0,3}\| \end{aligned}$$

This error bound is more than three times better at the cost of first-order continuity.

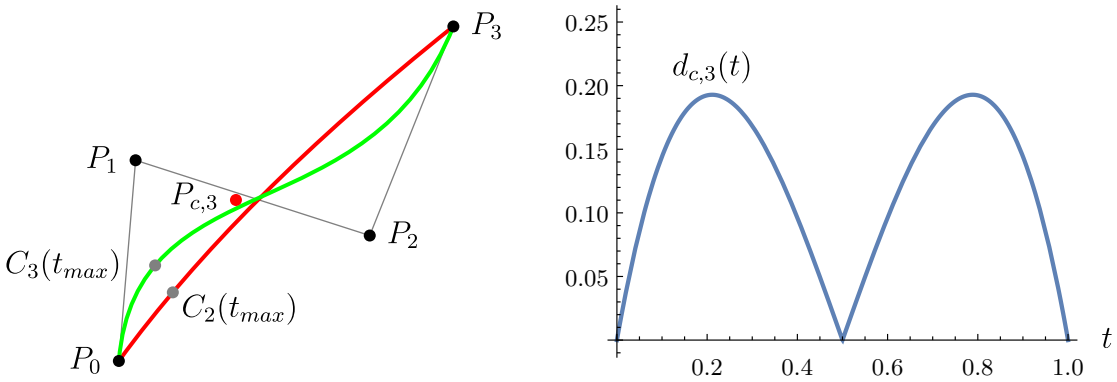

Figure 3.4: Parametric distance of a cubic curve (II)

3.4 Subdivision of cubic curves

Similar to the quadratic case, we will subdivide the cubic curve with blossoming to achieve a specific error limit across the whole curve. According to (2.5) the points are:

$$\begin{aligned} \breve{P}_0 &= P_0^{(3)}(t_a, t_a, t_a) \\ \breve{P}_1 &= P_0^{(3)}(t_a, t_a, t_b) \\ \breve{P}_2 &= P_0^{(3)}(t_a, t_b, t_b) \\ \breve{P}_3 &= P_0^{(3)}(t_b, t_b, t_b) \end{aligned}$$

Let $t_b = t_a + t_\Delta$ so that we get a curve segment of constant parameter step t_Δ and let k be a generic constant that represents the different factors from the previous section ($k = \frac{4}{27}$ for the first case and $k = \frac{1}{12\sqrt{3}}$ for the second one). Thus, inserting the points of the subcurve into $d_{max,3}$ yields:

$$\begin{aligned} d_{max,3} &= k\|\breve{P}_0 - 3\breve{P}_1 + 3\breve{P}_2 - \breve{P}_3\| \\ &= k\|P_0 - 3P_1 + 3P_2 - P_3\|t_\Delta^3 \\ &= k\|Q_{0,3}\|t_\Delta^3 \end{aligned}$$

Again, solving the expression for t_Δ gives

$$t_\Delta = \sqrt[3]{\frac{d_{max,3}}{k\|Q_{0,3}\|}} \tag{3.6}$$

which is the parameter interval to subdivide a cubic curve between $[t, t+t_\Delta]$ while also retaining the error bound $d_{max,3}$. We may choose between different parameterizations to approximate cubic curve segments by a single quadratic curve. Are we able to achieve a better result by approximating segments with two curves?

3.5 Alternative approximation of cubic curves

The drawback of the previous parameterizations is that we generally cannot achieve first-order continuity for both endpoints with just one curve. Therefore, the idea is to subdivide the cubic curve segment at $t = \frac{1}{2}$ so that the first half is parameterized for a continuous starting point and the second half for a continuous ending point. Due to this full continuity, the approximation is especially useful for offsetting curves (which will be discussed in an upcoming chapter). The points of the parameterization with the two quadratic curves $C_{2,a}(t)$ and $C_{2,b}(t)$ are defined by:

$$P_{c,a} = \left(1 - \tfrac{3}{4}\right) P_0 + \tfrac{3}{4} P_1$$
$$P_{c,b} = \left(1 - \tfrac{3}{4}\right) P_3 + \tfrac{3}{4} P_2$$
$$P_m = \left(1 - \tfrac{1}{2}\right) P_{c,a} + \tfrac{1}{2} P_{c,b}$$

The curves consist of $P_0, P_{c,a}, P_m$ and $P_m, P_{c,b}, P_3$. It is easy to show that

$$C_3(\tfrac{1}{2}) = \tfrac{1}{8}(P_0 - 3P_1 + 3P_2 - P_3) = P_m$$

and also

$$C'_{2,a}(1) = \tfrac{1}{4}(-P_0 - 3P_1 + 3P_2 + P_3) = C'_{2,b}(0)$$

which implies first-order continuity at the transition between $C_{2,a}$ and $C_{2,b}$. Furthermore, we can easily observe (see figure 3.5) that generally $C'_{2,a}(1) \neq C'(\frac{1}{2}) \neq C'_{2,b}(0)$.

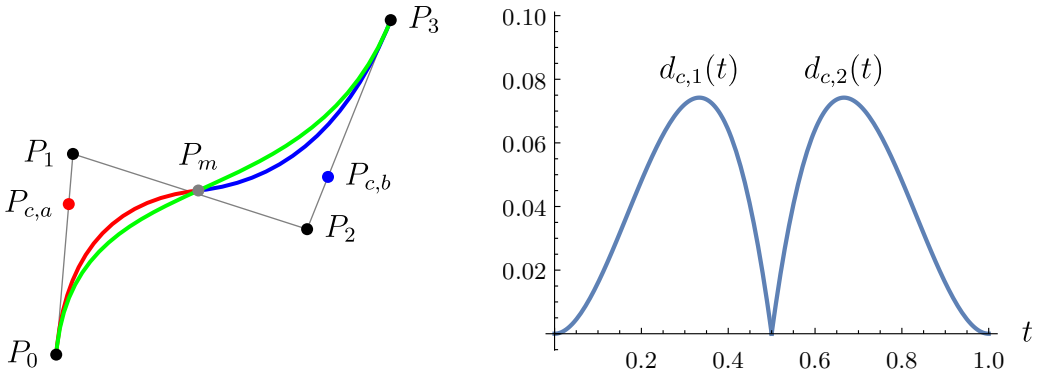

Figure 3.5: Parametric distance of a cubic curve (III)

To keep the same error bound $d_{max,3}$ we have to double (3.6) such that:

$$t_\Delta = 2\sqrt[3]{\frac{d_{max,3}}{k\|Q_{0,3}\|}} = \sqrt[3]{\frac{8\,d_{max,3}}{k\|Q_{0,3}\|}}$$

This means that $d_{max,3}$ may be decreased by a factor of 8 if the curve is halved. So for the current parameterization with two quadratic curves, the factor changes to $k = \frac{1}{54}$ and can be applied to (3.6).

If $P_0 = P_{c,a} = P_1$ or $P_2 = P_{c,b} = P_3$ we will come across a special case where the particular quadratic curve degenerates into a line segment. Mathematically, the first-order continuity is still given but we easily notice that the tangents of the cubic and the degenerate quadratic curve are different. Unfortunately, there is no obvious way to fix this without abandoning the first-order continuity at $C_{2,a}(1)$ and $C_{2,b}(0)$ as well as using only one quadratic curve. Therefore, we simply choose to leave it at that.

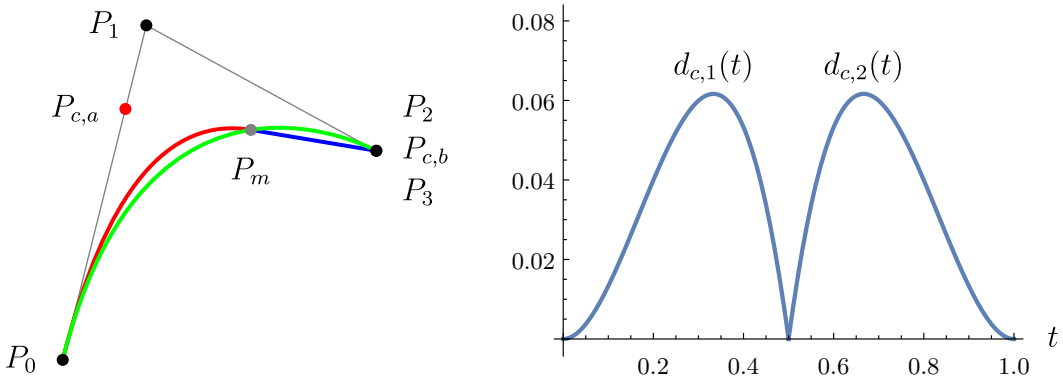

Figure 3.6: Degenerate case where $C_{2,b}(t)$ becomes a line segment

3.6 Maximum error of rational curves

Let $C_2(t)$ be a quadratic approximation of a rational curve $C_r(t)$ in standard form with $w > 0$. In this case all points of the curves (P_0, P_1 and P_2) are mutual:

$$C_r(t) = \frac{(1-t)^2 P_0 + 2t(1-t)wP_1 + t^2 P_2}{(1-t)^2 + 2t(1-t)w + t^2}$$

$$C_2(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

In *High order approximation of conic sections by quadratic splines* Floater has already learned that $\|C_r(t) - C_2(t)\|$ generally does not give satisfying results [3]. Instead, he proposed to reparameterize the quadratic curve with

$$s(t) = \frac{t(1 + (w-1)(1-t))}{1 + 2t(w-1)(1-t)}$$

so that $d_{2,r}(t) = \|C_r(t) - C_2(s(t))\|$ describes the parametric distance of the curves. Then symbolically solving $d'_{2,r}(t) = 0$ yields $t_{max} = \frac{1}{2}$ which gives the error bound of an approximated quadratic curve:

$$d_{max,r} = \tfrac{1}{4}\tfrac{|w-1|}{w+1}\|P_0 - 2P_1 + P_2\|$$

We notice that $d_{max,r}$ vanishes for $w \to 1$. This is plausible because with the same control points and $w = 1$ the curves $C_r(t)$ and $C_2(t)$ are equivalent.

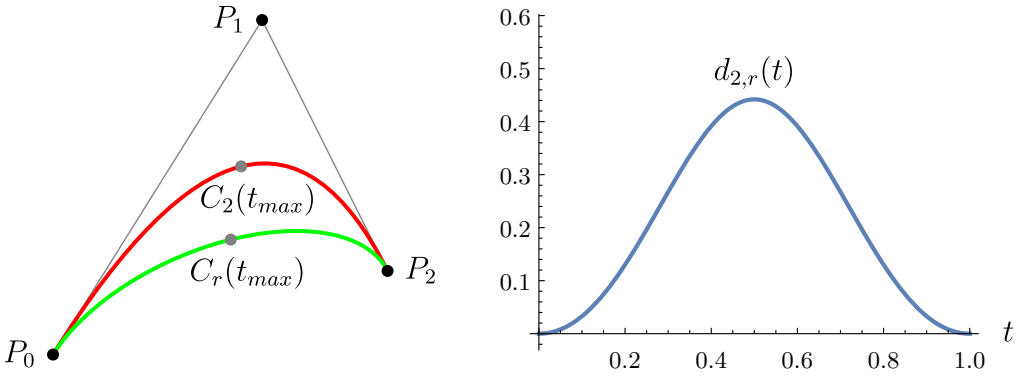

Figure 3.7: Parametric distance of a rational curve

3.7 Subdivision approximation of rational curves

Although the approach that Floater described in his work is recursive, we will discuss an incremental alternative. However, subdividing the curve with blossoming gives a very complex expression which is not convenient to solve anymore.
Therefore, we try to approximate the error bound by

$$d_{max,r} \approx \tfrac{1}{4}\tfrac{|w-1|}{w+1}\|P_0 - 2P_1 + P_2\|t_\Delta^\nu \tag{3.7}$$

such that ν roughly satisfies

$$d_{max,r} \gtrsim \max_{t_r \in [0,1]} \min_{t_2 \in [0,1]} \|\breve{C}_r(t_r) - \breve{C}_2(t_2)\|$$

for the subdivided curves $\breve{C}_2(t)$ and $\breve{C}_r(t)$ at $[t, t + t_\Delta]$ within a reasonable amount. This means that $d_{max,r}$ is not an upper bound and only supposed to be greater than the Hausdorff distance for the majority of cases (i.e. with occasional outliers). Then we would be able to solve (3.7) for t_Δ in a familiar manner:

$$t_\Delta = \left(\frac{4(w+1)\, d_{max,r}}{|w-1|\|P_0 - 2P_1 + P_2\|}\right)^{\frac{1}{\nu}}$$

Floater states that the error bound of the recursive method converges optimally with an order of four. Thus, we assume $\nu = 4$ and analyze the real error bound by measuring the directed Hausdorff distance.
The following data (see table 3.1) is taken from 1000 rational curves with randomly placed points in a grid of 100x100 and an error bound of $d_{max,r} = 0.05$. We distinguish between different ranges of weights and also measure the total number of subdivided segments. Each segment will be approximated by 8192 points to ensure accurate data for the minimum, average and maximum Hausdorff distance.

w_{min}	w_{max}	**Segments**	**Minimum**	**Average**	**Maximum**	**Outliers**
0.1	0.5	4332	0.00049	0.01489	0.04460	0
0.5	1	3135	0.00076	0.02543	0.05009	2
1	2	3356	0.00123	0.03189	0.06843	512
2	10	4582	0.00058	0.03366	0.11649	921

Table 3.1: Segment count and Hausdorff distance for $\nu = 4$

We notice that there are generally less segments needed in the vicinity of $w = 1$ and more for rather extreme weights. Furthermore, the maximum Hausdorff distance seems to increase together with w and clearly surpass the error limit, confirming that it is not an upper bound. Although the average distance is far beneath $d_{max,r}$ for all tested weights, up to 20 percent of the segments are above it.
It is possible to increase the precision of the approximation by slightly reducing ν. Table 3.2 shows the corresponding values for $\nu = 3.5$ which include a higher segment count and lower Hausdorff distances. For $w_{min} = 2$ and $w_{max} = 10$ there still exist outliers that exceed the error bound.

w_{min}	w_{max}	**Segments**	**Minimum**	**Average**	**Maximum**	**Outliers**
0.1	0.5	5178	0.00012	0.00709	0.02417	0
0.5	1	3567	0.00052	0.01390	0.04944	0
1	2	3869	0.00027	0.01600	0.04814	0
2	10	5466	0.00026	0.01592	0.05807	61

Table 3.2: Segment count and Hausdorff distance for $\nu = 3.5$

Remark. The simulated data demonstrates that is it difficult to assure a specific error limit for the incremental approach of rational curves, as opposed to the quadratic and cubic case. If this guarantee was required we would suggest using the recursive method described by Floater [3].

Chapter 4

Flattening

This chapter covers the definition of flatness and provides algorithms to approximate quadratic, cubic and rational curves by line segments. This process is commonly referred to as *flattening*. The general approach is to simplify cubic and rational curves to quadratic ones first. Finally, all quadratic curves are simplified one step further so that they are effectively flattened.

4.1 Flatness

Flatness is usually described as some vaguely defined distance between a curve and its approximated line segment. In the following, we will define flatness with the help of the Hausdorff distance. Although its calculation is expensive (and possibly not suitable for real-time graphics) it provides the most precise information.
Similar to the approximation error, we define flatness as the directional Hausdorff distance between an arbitrary curve C_n and a linear curve C_1:

$$f_n = \max_{t_n \in [0,1]} \min_{t_1 \in [0,1]} \|C_1(t_1) - C_n(t_n)\|$$

In the previous chapter we have found multiple error limits for simplifying curves which may be used as a bound for flatness. So with (2.8) the flatness of a quadratic curve is bounded by:

$$\begin{aligned} f_2 &= \max_{t_2 \in [0,1]} \min_{t_1 \in [0,1]} \|C_1(t_1) - C_2(t_2)\| \\ &\leq d_{max,2} \end{aligned}$$

For cubic curves we limit the flatness criterion by also adding the error bound of the

degree reduction from a cubic curve to a quadratic one:

$$f_3 = \underbrace{\max_{t_3\in[0,1]} \min_{t_2\in[0,1]} \|C_2(t_2) - C_3(t_3)\|}_{\leq d_{max,3}} + \underbrace{\max_{t_2\in[0,1]} \min_{t_1\in[0,1]} \|C_1(t_1) - C_2(t_2)\|}_{\leq d_{max,2}}$$
$$\leq d_{max,3} + d_{max,2}$$

Similarly, the flatness for a rational curve is bounded by

$$f_r = \underbrace{\max_{t_r\in[0,1]} \min_{t_2\in[0,1]} \|C_2(t_2) - C_r(t_r)\|}_{\leq d_{max,r}} + \underbrace{\max_{t_2\in[0,1]} \min_{t_1\in[0,1]} \|C_1(t_1) - C_2(t_2)\|}_{\leq d_{max,2}}$$
$$\leq d_{max,r} + d_{max,2}$$

which includes the approximation error of simplifying the curve to a quadratic one. *Remark.* The relation $f_r \leq d_{max,r} + d_{max,2}$ does not strictly hold for incremental subdivisions of the curve, as pointed out in section 3.6.

4.2 Flattening of curves

Since the flatness criteria are limited by the maximum parametric distance, we are able to utilize the previously developed simplification methods for flattening cubic, quadratic and rational curves incrementally.
The flattening of a quadratic curve is directly given in section 3.2 because the generated linear curves are basically line segments. We will describe the method in the form of pseudocode as follows:

procedure FLATTENQUADRATICCURVE($P, C_2, d_{max,2}$)
  Set output path P
  Set quadratic curve C_2
  Set flatten tolerance $d_{max,2}$
  Calculate t_Δ from $d_{max,2}$ and C_2
  for all $t \in \{0, t_\Delta, 2t_\Delta, \ldots, 1\}$ **do**
    Evaluate $C_2(t)$ and add line segment to P
  end for
end procedure

To flatten cubic and rational curves we will simplify them to quadratic curve segments and then apply *FlattenQuadraticCurve* to each segment.
There are two appropriate parameterizations to choose for the simplification of cubic curves. The first parameterization is the single curve midpoint approximation from

section 3.3 ($k = \frac{1}{12\sqrt{3}}$). The second one was mentioned in section 3.5 and approximates a cubic curve segment by two quadratic ones ($k = \frac{1}{54}$). Both seem equally viable in terms of quality so that benchmarks need to show if one is better than the other.
The pseudocode for the cubic curves is covered below:

procedure FLATTENCUBICCURVE($P, C_3, d_{max,3}, d_{max,2}$)
  Set output path P
  Set cubic curve C_3
  Set simplify tolerance $d_{max,3}$
  Set flatten tolerance $d_{max,2}$
  Calculate t_Δ from $d_{max,3}$ and C_3
  for all $[t_a, t_b] \in \{[0, t_\Delta], [t_\Delta, 2t_\Delta], \ldots, [\ldots, 1]\}$ **do**
    Subdivide C_3 between $[t_a, t_b]$ and set $\breve{C}_3$
    Approximate $\breve{C}_3$ by $\breve{C}_2$
    FLATTENQUADRATICCURVE($P, \breve{C}_2, d_{max,2} - d_{max,3}$)
  end for
end procedure

For good results it is suggested that $d_{max,3} < d_{max,2}$ as $d_{max,3}$ controls how much the flattened line segments are displaced from the original curve. This means that it is generally not possible to generate points on the original cubic curve. However, observations show that $d_{max,3} = \frac{1}{4}\, d_{max,2}$ gives a balanced visual impression.
Similarly, the pseudocode for rational curves is given by:

procedure FLATTENRATIONALCURVE($P, C_r, d_{max,r}, d_{max,2}$)
  Set output path P
  Set rational curve C_r
  Set simplify tolerance $d_{max,r}$
  Set flatten tolerance $d_{max,2}$
  Calculate t_Δ from $d_{max,r}$ and C_r
  for all $[t_a, t_b] \in \{[0, t_\Delta], [t_\Delta, 2t_\Delta], \ldots, [\ldots, 1]\}$ **do**
    Subdivide C_r between $[t, t + t_\Delta]$ and set $\breve{C}_r$
    Approximate $\breve{C}_r$ by $\breve{C}_2$
    FLATTENQUADRATICCURVE($P, \breve{C}_2, d_{max,2} - d_{max,r}$)
  end for
end procedure

Just as the cubic case, the simplification leads to a displacement of the computed line segments. Thus, the same ratio of $d_{max,r} = \frac{1}{4}\, d_{max,2}$ is recommended.

4.3 Performance optimizations

Because of the incremental approach there are ways to optimize the performance of the algorithms that are presented in the previous section.
The evaluation of $C_2(t)$ in the inner loop of *FlattenQuadraticCurve* may be improved by calculating the coefficients $Q_{0,2}, Q_{1,2}$ and $Q_{2,2}$ beforehand and then use Horner's method to actually evaluate

$$C_2(t) = Q_{2,2} + t(Q_{1,2} + tQ_{0,2})$$

which minimizes the arithmetic operations. Performance can be enhanced even further by vectorization through specialized CPU instruction sets (e.g. SSE or AVX) so that multiple values are computed in parallel.
To improve the subdivision portion of *FlattenCubicCurve* and *FlattenRationalCurve* we will first look at the reparameterization of t_Δ. Let $t_\Delta^{(k)}$ be the current parameter interval that needs to be scaled between the curve segment $[t_a, t_b]$. Then

$$t_\Delta^{(k+1)} = \frac{t_\Delta^{(k)}}{(t_b - t_a) - t_\Delta^{(k)}}$$

for $\{t_a, t_b\} \in [0, 1]$ and $0 \leq t_\Delta^{(k)} \leq t_b - t_a$ is the scaled parameter interval. Now, instead of subdividing curves between $\{[0, t_\Delta], [t_\Delta, 2t_\Delta], \ldots, [\ldots, 1]\}$ we iteratively subdivide $[0, t_\Delta]$ and reparameterize t_Δ with $t_b - t_a = 1$ until $t_\Delta \geq 1$. This is better because subdividing $[0, t_\Delta]$ and rescaling t_Δ generally needs less arithmetic operations than subdividing in the middle of a curve.
Additionally, it is also possible to utilize the reparameterization for multi-threading. E.g. for n tasks we could subdivide a curve into n pieces of $\{[0, n_\Delta], [n_\Delta, 2n_\Delta], \ldots, [(n-1)n_\Delta, 1]\}$ with $n_\Delta = \lfloor \frac{1}{t_\Delta n} \rfloor t_\Delta$ and $t_b - t_a = \frac{1}{n}$. The question of how to optimally arrange and schedule the tasks remains open though.

4.4 Proposal: Alternative approaches

While writing this thesis there were some ideas for approximating cubic and rational curves directly with line segments. The assumption is that omitting the simplification step could result in an overall faster approach. However, the following methods are not complete and expose some problems that have not been solved yet. They should be understood as a proposal and need further investigation.
First, we will look at the maximum parametric distance of a single cubic and rational curve to get the error bound for the flatness and then analyze the variations of the parametric distance over the course of the curve.

The parametric distances are defined by:

$$d_{max,3} = \max_{t\in[0,1]} d_{1,3}(t)$$

$$d_{max,r} = \max_{t\in[0,1]} d_{1,r}(t)$$

Now, let the two curves

$$C_{2,a}(t) = (1-t)^2 P_0 + 2t(1-t)P_{c,a} + t^2 P_3$$
$$C_{2,b}(t) = (1-t)^2 P_0 + 2t(1-t)P_{c,b} + t^2 P_3$$

with the control points

$$P_{c,a} = \left(1 - \tfrac{3}{2}\right) P_0 + \tfrac{3}{2} P_1$$
$$P_{c,b} = \left(1 - \tfrac{3}{2}\right) P_3 + \tfrac{3}{2} P_2$$

be a decomposition of $C_3(t)$ such that:

$$C_3(t) = (1-t)\, C_{2,a}(t) + t C_{2,b}(t)$$

With the approximated linear curve

$$C_1(t) = (1-t)P_0 + tP_3$$

we define the parametric distances for the two curves $C_{2,a}(t)$ and $C_{2,b}(t)$ as:

$$d_a(t) = \|C_{2,a}(t) - C_1(t)\|$$
$$d_b(t) = \|C_{2,b}(t) - C_1(t)\|$$

Then we interpolate $d_a(t)$ and $d_b(t)$ to get the approximated parametric distance

$$\tilde{d}_{1,3}(t) = \left(1 - \tfrac{1}{2}\right) d_a(t) + \tfrac{1}{2} d_b(t)$$

which has its maximum at $t_{max} = \frac{1}{2}$. Thus, we suggest that

$$\tilde{d}_{max,3} = \tfrac{1}{8}\|2P_0 - 3P_1 + P_3\| + \tfrac{1}{8}\|P_0 - 3P_2 + 2P_3\|$$

is a good approximation of the error limit (not proven to be an upper bound).

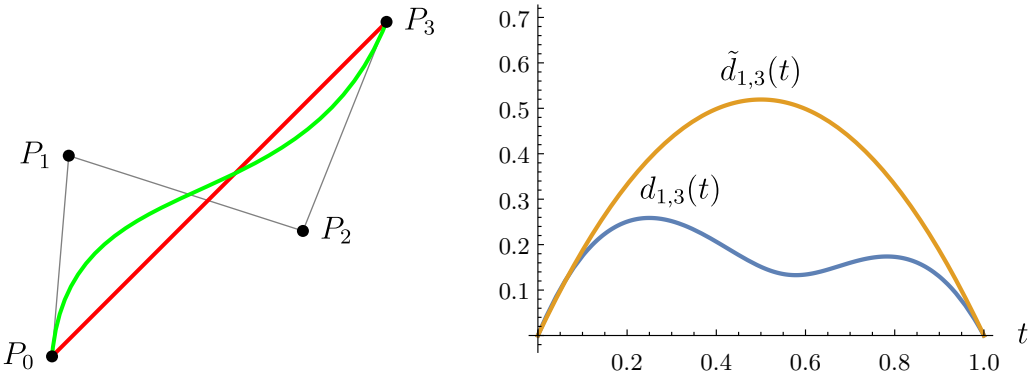

Figure 4.1: Parametric distance of a cubic curve

Furthermore, with

$$C_r(t) = \frac{(1-t)^2 P_0 + 2t(1-t)wP_1 + t^2 P_2}{(1-t)^2 + 2t(1-t)w + t^2}$$

$$C_1(t) = (1-t)P_0 + tP_2$$

and $w > 0$ we are able to obtain the maximum of the parametric distance

$$d_{1,r}(t) = \|C_r(t) - C_1(s(t))\|$$

similar to section 3.6. Again, the maximum is found at $t_{max} = \frac{1}{2}$ so that

$$d_{max,r} = \tfrac{1}{2}\tfrac{w}{w+1}\|P_0 - 2P_1 + P_2\|$$

is the error bound.

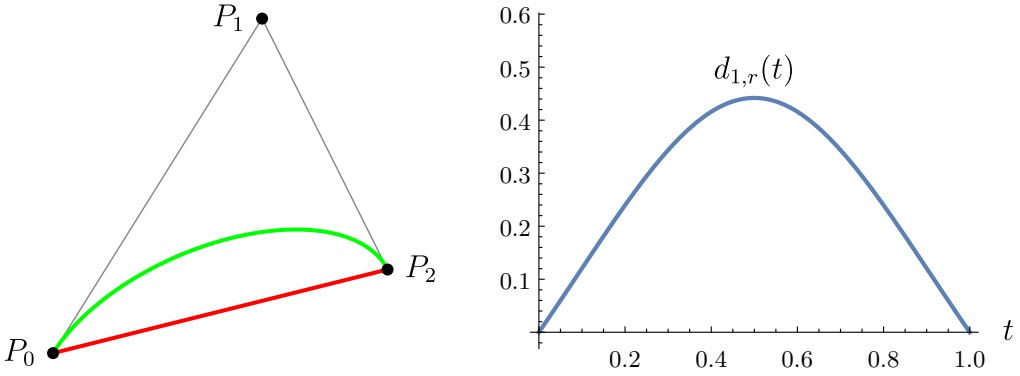

Figure 4.2: Parametric distance of a rational curve

Unlike quadratic curves, the parameter interval t_Δ cannot be uniformly spaced across cubic and rational curves so that it keeps a constant parametric distance. Thus, the general idea would be to find transformations $t_n(t)$ that are able to distribute scaled curve segments $[t_n(t), t_n(t + t_\Delta)]$ which retain the specified error limit.
To analyze this issue we define the momentary parametric distance as:

$$d_n(t, t_\Delta) = \|C_n(t) - \tfrac{1}{2}C_n(t - \tfrac{1}{2}t_\Delta) - \tfrac{1}{2}C_n(t + \tfrac{1}{2}t_\Delta)\|$$

For quadratic curves we obtain the same result as in section 3.2:

$$\begin{aligned} d_2(t, t_\Delta) &= \tfrac{1}{4}\|P_0 - 2P_1 + P_2\|t_\Delta^2 \\ &= \tfrac{1}{4}\|Q_{0,2}\|t_\Delta^2 \end{aligned}$$

If we apply the distance function to cubic curves (and reorder the expression) we will get an interesting result:

$$\begin{aligned} d_3(t, t_\Delta) &= \tfrac{3}{4}\|(P_0 - 2P_1 + P_2) - t(P_0 - 3P_1 + 3P_2 - P_3)\|t_\Delta^2 \\ &= \tfrac{3}{4}\|\tfrac{1}{3}Q_{1,3} - tQ_{0,3}\|t_\Delta^2 \end{aligned}$$

Contrary to the quadratic case, the distance is dependent of t. Furthermore, we are able to calculate the minimum of the expression for $t_\Delta = 1$:

$$t_{min} = \frac{Q_{0,3} \cdot Q_{1,3}}{3(Q_{0,3} \cdot Q_{0,3})}$$

To adjust the parametric distance for $C_3(t)$ it is theorized that the parameter interval t_Δ would have to grow or shrink between $[0, t_{min}]$ and then change back in the other direction between $[t_{min}, 1]$. How to actually obtain such a transformation $t_3(t)$ is left as an open problem for now.

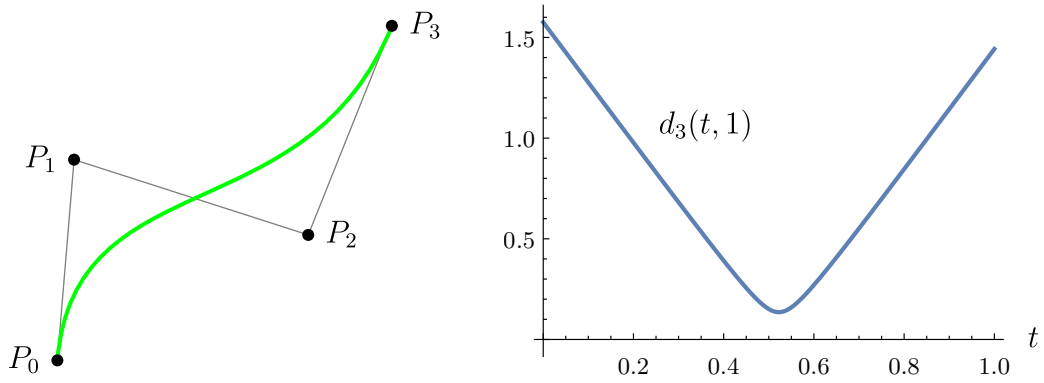

Figure 4.3: Parametric distance variation of a cubic curve

Unfortunately, the expression for $d_r(t, t_\Delta)$ cannot be presented in a meaningful way because it is verbose and hard to reorder. We may deduce from the graph in figure 4.4 that $d_r(t, t_\Delta)$ is in fact not a simple expression.

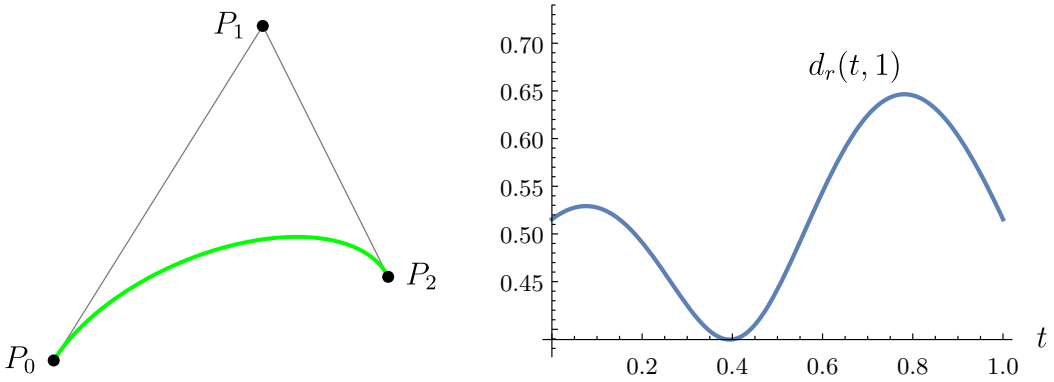

Figure 4.4: Parametric distance variation of a rational curve

Despite the complexity we assume that a proper reparameterization of the curve could help in this case, i.e. similar to $s(t)$ in section 3.6. However, due to time restraints none of such has been attempted.

Chapter 5

Offsetting

In the following chapter we will investigate offset curves and their geometry to further introduce algorithms for *offsetting* quadratic curves, i.e. approximating parallel curves. Just like the flattening approach, cubic and rational curves are going to be simplified to quadratic ones first so that, ultimately, all cases can be covered by the methods for offsetting quadratic curves.

5.1 Offset curves

An offset curve of a curve $C(t)$ is defined by $\bar{C}(t) = C(t) + \bar{d}\vec{n}(t)$ with the offset distance $\bar{d}$ and the unit normal $\vec{n}(t)$ of the derivative $C'(t)$ such that:

$$\vec{n}_x(t) = +\frac{C'_y(t)}{\sqrt{C'_x(t)^2 + C'_y(t)^2}} \qquad \vec{n}_y(t) = -\frac{C'_x(t)}{\sqrt{C'_x(t)^2 + C'_y(t)^2}}$$

Because of the square roots in the denominators, offset curves generally cannot be expressed by the same type of curve and therefore have to be approximated instead.

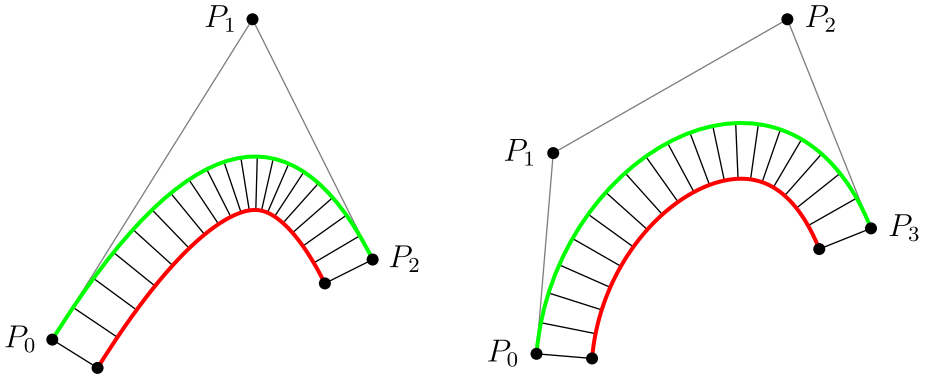

Figure 5.1: Quadratic and cubic offset curves

Remark. There exists a special type of curve which satisfies $C'_x(t)^2 + C'_y(t)^2 = \sigma(t)^2$ and is able to be precisely represent its offset curve by a rational one. These curves are called Pythagorean-hodograph curves and were extensively studied by Farouki [20].

5.2 Offsetting of curves

In *Precise offsetting of quadratic Bézier curves* we have already described a method to approximate quadratic curves by one or multiple quadratic curve segments [13]. This is convenient because offsetting cubic and rational curves can be reduced to the quadratic case as well. Since the approach does not immediately flatten the curve it is still necessary to apply flattening to all quadratic curve segments, e.g. with the previous algorithm *FlattenQuadraticCurve* from section 4.2.
To obtain the points $\bar{P}_0, \bar{P}_1$ and $\bar{P}_2$ of the approximated offset curve, the edges $\overline{P_1P_0}$ and $\overline{P_2P_1}$ of the control polygon are translated by the offset distance $\bar{d}$ so that:

$$\vec{n}_0 = \vec{n}(0) \qquad \vec{n}_1 = \vec{n}(1) \qquad \vec{n} = \vec{n}_0 + \vec{n}_1$$

$$\bar{P}_0 = P_0 \pm \bar{d}\vec{n}_0 \qquad \bar{P}_2 = P_2 \pm \bar{d}\vec{n}_1 \qquad \bar{P}_1 = P_1 \pm \frac{2\bar{d}\vec{n}}{\vec{n} \cdot \vec{n}}$$

Remark. The denominator of the translation vector in $\bar{P}_1$ is a dot product of $\vec{n}$.

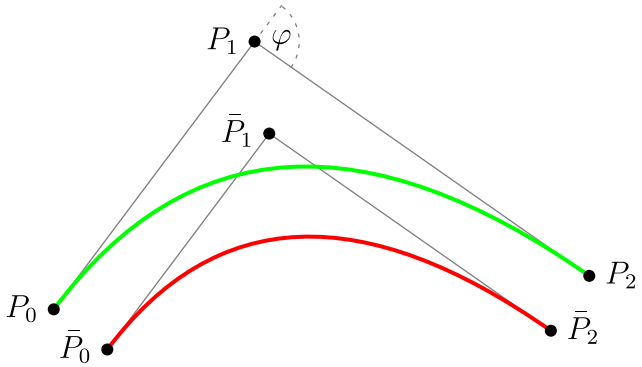

Figure 5.2: Approximated offset of a quadratic curve

The error bound depends on the offset distance $\bar{d}$ as well as the angle φ between $\overline{P_1P_0}$ and $\overline{P_2P_1}$. It is defined by

$$\bar{d}_{max,2} = \frac{\bar{d}}{4}(3 + \cos(\varphi)) \sec\left(\frac{\varphi}{2}\right)$$

with sec() as the secant function. A specific error limit may be bounded across the curve by iteratively subdividing it into segments with a maximum angle of φ.

Additionally, the paper elaborates on how to further improve the precision of the offset approximation and unites all those techniques into an algorithm which will be referred to as *OffsetQuadraticCurve*. To use this method with cubic and rational curves, we need to simplify them to quadratic ones and subsequently apply *OffsetQuadraticCurve* for each curve segment.
Although the relevant algorithms are very similar to the ones mentioned in section 4.2, we will present them below for the purpose of clarity. The pseudocode for offsetting cubic curves is described as follows:

procedure OFFSETCUBICCURVE($P, C_3, d_{max,3}, \bar{d}, \varphi$)
  Set output path P
  Set cubic curve C_3
  Set simplify tolerance $d_{max,3}$
  Set offset distance $\bar{d}$
  Set subdivision angle φ
  Calculate t_Δ from $d_{max,3}$ and C_3
  for all $[t_a, t_b] \in \{[0, t_\Delta], [t_\Delta, 2t_\Delta], \ldots, [\ldots, 1]\}$ **do**
    Subdivide C_3 between $[t_a, t_b]$ and set $\breve{C}_3$
    Approximate $\breve{C}_3$ by $\breve{C}_{2,a}$ and $\breve{C}_{2,b}$
    OFFSETQUADRATICCURVE($P, \breve{C}_{2,a}, \bar{d}, \varphi$)
    OFFSETQUADRATICCURVE($P, \breve{C}_{2,b}, \bar{d}, \varphi$)
  end for
end procedure

Finally, the pseudocode for rational curves is given by:

procedure OFFSETRATIONALCURVE($P, C_r, d_{max,r}, \bar{d}, \varphi$)
  Set output path P
  Set rational curve C_r
  Set simplify tolerance $d_{max,r}$
  Set offset distance $\bar{d}$
  Set subdivision angle φ
  Calculate t_Δ from $d_{max,r}$ and C_r
  for all $[t_a, t_b] \in \{[0, t_\Delta], [t_\Delta, 2t_\Delta], \ldots, [\ldots, 1]\}$ **do**
    Subdivide C_r between $[t, t + t_\Delta]$ and set $\breve{C}_r$
    Approximate $\breve{C}_r$ by $\breve{C}_2$
    OFFSETQUADRATICCURVE($P, \breve{C}_2, \bar{d}, \varphi$)
  end for
end procedure

5.3 Curvature and cusps

The local expression of the (signed) curvature is defined by

$$\kappa = \frac{\det(\gamma', \gamma'')}{\|\gamma'\|^3}$$

with γ being a twice differentiable parametric curve $C(t)$. This translates to

$$\kappa(t) = \frac{C'(t) \times C''(t)}{\|C'(t)\|^3}$$

for planar curves where $\times$ denotes the two-dimensional cross product. The curvature radius is simply the reciprocal, thus:

$$R(t) = \frac{1}{\kappa(t)} = \frac{\|C'(t)\|^3}{C'(t) \times C''(t)} \tag{5.1}$$

Cusps are singular points of a curve. The necessary condition [21] for such points is that the first derivative vector of the curve vanishes:

$$C'(t) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Applying the euclidean distance and inserting it in (5.1) yields:

$$\|C'(t)\| \to 0 \implies R(t) \to 0$$

So the condition for the location of a cusp is $R(t) = 0$ with parameter $t \in [0, 1]$. This equation is useful because it allows to check for a range of curvature radius in the vicinity of the cusp, e.g. by solving $R(t) = R_\Delta$ for (real) $t \in [0, 1]$.

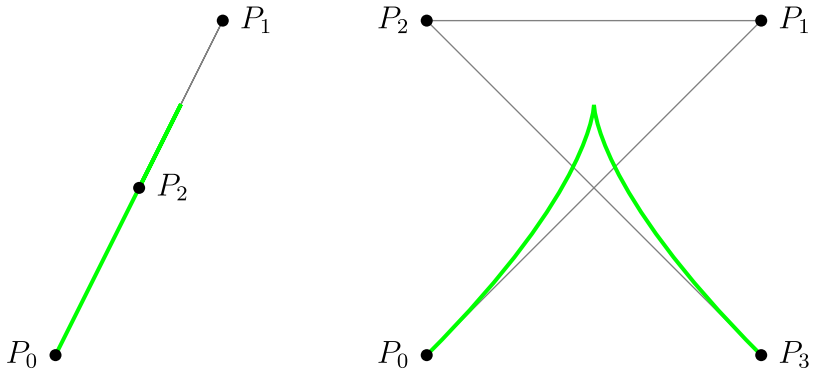

Figure 5.3: Cusps of quadratic and cubic curves

Linear curves cannot have any cusps because $\deg(C_1) = 0$ and quadratic curves may only have one cusp at a time. However, cubic curves can have up to five curvature extrema [22] and a theoretical maximum of four cusps.

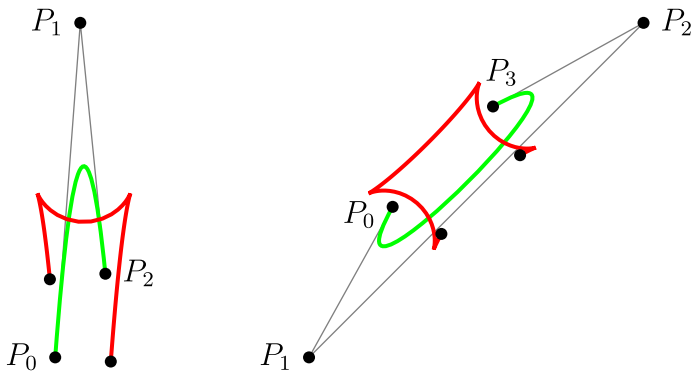

Figure 5.4: Cusps of quadratic and cubic offset curves

We may observe in figure 5.4 that for every emerging cusp of a curve $C(t)$ there might be two cusps in the offset curve $\bar{C}(t)$. If $C(t)$ gets close to the cusp case, the interval between the two cusps in $\bar{C}(t)$ will slowly turn into a circular segment. In the exact cusp case the derivative $C'(t)$ vanishes and the offset path can be considered defective because the offset curve cannot the evaluated at the location of the cusp. This is a pathological case which needs to be recognized and rectified.
Fortunately, we only need to consider quadratic curves, i.e. after cubic and rational curves have been simplified. So checking the angle between $\overline{P_1P_0}$ and $\overline{P_1P_2}$ is a simple method to test for an imminent cusp. Unlike solving $R(t) = R_\Delta$ for every curve, the angle check is resolution independent and might be implemented more efficiently.

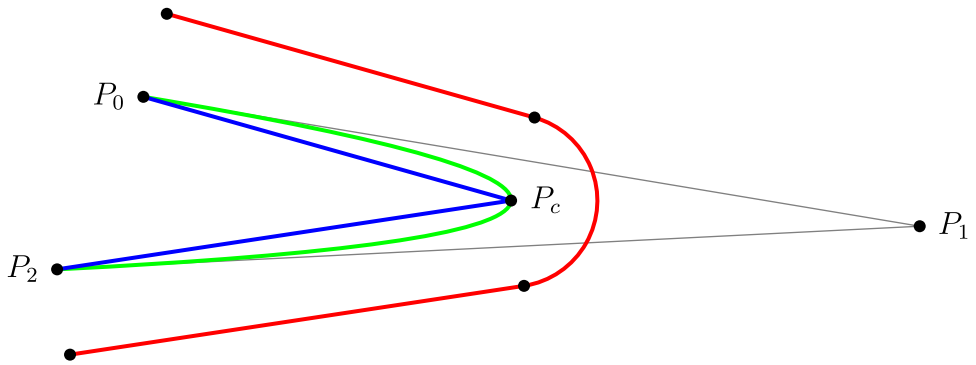

Figure 5.5: Fixing the cusp of a quadratic curve

If the angle dropped beneath a specified threshold we could fix the pathological case by approximating the offset curve with two line segments and an appropriate circular segment in between, as shown in figure 5.5. The line segments are connected by the point P_c which is the point of the minimal curvature. It may be easily obtained by

symbolically solving $R'(t) = 0$ for (real) t. This yields

$$t_c = -\frac{Q_{0,2} \cdot Q_{1,2}}{2(Q_{0,2} \cdot Q_{0,2})}$$

which is the parametric location of the cusp such that $P_c = C_2(t_c)$. It should be noted that the circular segment can be represented by one or more rational curves [16].

5.4 Proposal: Immediate flattening

If we are willing to sacrifice the error limit assurance of the previously presented method in favor of performance, it was theorized that the immediate flattening of a quadratic offset curve might be faster than flattening it afterwards. Therefore, we will outline an experimental approach that uses circular approximation which is similar to Hain's method [8]. But, instead of approximating the flatness criterion with an estimated curvature radius, we will use the approximation from section 3.2 and simply calculate the unsigned curvature radius at the current point along the curve.

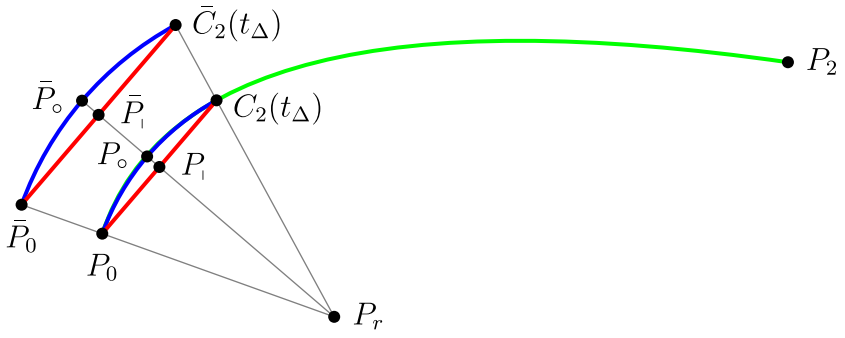

Figure 5.6: Circular approximation of a quadratic curve

For sufficiently small t_Δ we assume that $R(t) \approx R(t+t_\Delta)$ to establish the outer circular approximation and its relations, as shown in figure 5.6. The general goal is to find the parameter t_Δ for the current t where a specified flatness criterion $\bar{f}_2$ is met in the outer offset curve and then iteratively process the remaining curve. The inner line segment will always be overapproximated if it is generated from the same parameter. For the circular approximation the two flatness criteria f_2 and $\bar{f}_2$ may be approximated by the error bound $d_{max,2}$ and its counterpart $\bar{d}_{max,2}$ from the offset curve:

$$f_2 = \|P_\circ - P_1\| \approx d_{max,2}$$
$$\bar{f}_2 = \|\bar{P}_\circ - \bar{P}_1\| \approx \bar{d}_{max,2}$$

Furthermore, the triangles $\triangle P_\circ P_| P_0$ and $\triangle \bar{P}_\circ \bar{P}_| \bar{P}_0$ are similar to $\triangle P_| P_0 P_r$ and $\triangle \bar{P}_| \bar{P}_0 P_r$ so that the following ratios apply:

$$\frac{\bar{f}_2}{f_2} = \frac{\|\bar{P}_\circ - \bar{P}_|\|}{\|P_\circ - P_|\|} = \frac{\|\bar{P}_| - \bar{P}_0\|}{\|P_| - P_0\|} = \frac{\|\bar{P}_0 - P_r\|}{\|P_0 - P_r\|} = \frac{|R(t)| + \bar{d}}{|R(t)|}$$

We replace the flatness criteria with the corresponding maximum parametric distances and solve the equation for $d_{max,2}$ to obtain:

$$\frac{\bar{d}_{max,2}}{d_{max,2}} \approx \frac{|R(t)| + \bar{d}}{|R(t)|} \iff d_{max,2} = \bar{d}_{max,2} \frac{|R(t)|}{|R(t)| + \bar{d}}$$

In section 3.2 the correlation between $d_{max,2}$ and t_Δ was defined as:

$$t_\Delta = \sqrt{\frac{4\, d_{max,2}}{\|Q_{0,2}\|}}$$

Now, we can insert $d_{max,2}$ such that

$$t_\Delta = \sqrt{\frac{4\, \bar{d}_{max,2}}{\|Q_{0,2}\|} \frac{|R(t)|}{|R(t)| + \bar{d}}}$$

which is the parameter interval to approximate the next point on the offset curve. It needs to be recalculated iteratively because the curvature radius $R(t)$ changes for every step. At the current parameter t of the curve, the terms of $C_2(t), C_2'(t)$ and $C_2''(t)$ may be used for calculating the point $\bar{C}_2(t) = C_2(t) + \bar{d}\vec{n}(t)$ on the offset curve as well as the curvature radius $R(t)$ so that less arithmetic operations might be needed overall. The approach is described in the form of pseudocode as follows:

procedure OffsetAndFlattenQuadraticCurve($P, C_2, \bar{d}_{max,2}$)
    Set output path P
    Set quadratic curve C_2
    Set offset flatten tolerance $\bar{d}_{max,2}$
    Set $t = 0$
    repeat
        Evaluate and set $C_2(t), C_2'(t), C_2''(t)$
        Evaluate $\bar{C}_2(t)$ from $C_2(t), C_2'(t)$ and add line segment to P
        Calculate $|R(t)|$ from $C_2'(t), C_2''(t)$
        Calculate t_Δ from $|R(t)|, d_{max,2}$ and set $t = t + t_\Delta$
    until $t \geq 1$
    Evaluate $\bar{C}_2(t)$ with $t = 1$ and add line segment to P
end procedure

For sharp turns, e.g. as seen in figure 5.4, the presented method degrades similar to Hain's one because both try to assume a specific curvature radius over an estimated parameter interval. So if the radius deviates more quickly (which is generally the case for sharp turns) or even changes direction, the offset path may be defective. Unfortunately, Hain did not provide a suggestion on how to deal with such pathological cases in his paper. Therefore, we will suggest the angle check from the previous section as a basic test. Another improvement could be achieved by predicting the course of the curvature radius $R(t)$ to obtain a better approximation.

Remark. The technique of immediately flattening offset curves to line segments could possibly be applied to cubic and rational curves as well. However, the general problem of reparameterizing those curves (as mentioned in section 4.4) still persists and would have to be solved first.

Chapter 6

Run-time performance

Finally, we will compare the run-time performance of the developed approaches to currently established ones. All tests have been implemented with .NET Core 3.0 (C# 8.0), BenchmarkDotNet (0.11.5) and without optimizations in the form of multi-threading or vectorization. Furthermore, the tests ran on an AMD Ryzen 3700X CPU with Windows 10 (1909).
The methods will be applied to a path of N quadratic, cubic or rational curves whose points are randomly placed in a 100x100 grid. There are less cubic curves to balance out the total number of points in the path. The weights of the rational curves are arbitrarily chosen such that $0.5 \leq w \leq 2$.

6.1 Flattening

In the following we are going to measure the timings of the incremental approaches from section 4.2, Hain's flattening technique [7] and a recursive implementation that approximates flatness with the maximum deviation of the control points.

Curve	Method	N	Points	Mean	Error	StdDev
Quadratic	Incremental	1000	10680	78.46 us	0.1070 us	0.1001 us
	Recursive	1000	10068	369.72 us	0.1378 us	0.1289 us
	Hain	1000	7311	217.65 us	0.0992 us	0.0928 us
Cubic	Incremental 1	667	11257	304.10 us	0.1651 us	0.1545 us
	Incremental 2	667	12387	316.60 us	0.1151 us	0.1076 us
	Recursive	667	9409	483.40 us	0.2430 us	0.2273 us
	Hain	667	6808	331.90 us	0.1809 us	0.1692 us
Rational	Incremental	1000	12757	301.10 us	0.1944 us	0.1819 us
	Recursive	1000	9883	428.60 us	0.1859 us	0.1739 us

Table 6.1: Flattening of quadratic, cubic and rational curves

Table 6.1 shows the time needed to flatten the path as well as its total point count after the flattening. The flatness criterion is set to 0.25 which translates to $d_{max,2} = 0.2$ and $d_{max,3} = d_{max,r} = 0.05$ for the incremental method. As for the cubic case, we are testing the two parameterizations *Incremental 1* ($k = \frac{1}{12\sqrt{3}}$) and *Incremental 2* ($k = \frac{1}{54}$) which has already been mentioned in section 4.2.
The quadratic incremental method is 2.8 times faster than Hain's and 4.7 times faster than the recursive one which is a big gain of performance. For the cubic and the rational case the timings of the incremental approaches get closer to the other ones since the extra simplification step has a significant cost. Furthermore, we notice that *Incremental 1* is marginally better than *Incremental 2* in both performance and line count. The incremental approach is generally faster than both Hain's and the recursive one but we are paying for it with a relatively high line segment count which might cause additional overhead in the rasterization step. The flattening performance of the incremental method could be enhanced even further with the help of vectorization.

6.2 Offsetting

For offsetting we will compare between the incremental (and partly iterative) method discussed in section 5.2, the experimental approach *Dynamic* from section 5.4, the offset technique described by Hain [8] and the recursive subdivision method from AGG (Anti-Grain Geometry) [23].

Curve	Method	N	Points	Mean	Error	StdDev
Quadratic	Incremental	1000	30313	998.2 us	0.5397 us	0.5048 us
	Dynamic	1000	25326	548.7 us	0.1478 us	0.1382 us
	Hain	1000	17782	753.9 us	0.1521 us	0.1422 us
	AGG	1000	41856	1725.1 us	0.6888 us	0.6443 us
Cubic	Incremental	667	31925	1855.0 us	1.2530 us	1.1107 us
	Dynamic	667	27856	980.5 us	0.5613 us	0.5251 us
	Hain	667	15880	791.7 us	0.3029 us	0.2834 us
	AGG	667	42838	2393.6 us	1.5823 us	1.4027 us
Rational	Incremental	1000	33148	1577.9 us	0.4326 us	0.3835 us
	Dynamic	1000	28778	753.9 us	0.2782 us	0.2602 us

Table 6.2: Offsetting of quadratic, cubic and rational curves

The path will be processed by a stroker with an offset distance $\bar{d} = 0.5$ to both sides. *Incremental* uses an angle criterion of $\varphi = 22.5$ degrees such that the error bound is about 0.000188 for the iterative quadratic offset approximation which is a negligible amount for this test. The other criteria are the same as in the previous section.

The timings in table 6.2 attest a significant performance gain for the dynamic methods. They are almost twice as fast as the incremental ones, demonstrating that immediate flattening can achieve significant performance gains, however, at the loss of the strict precision control. Although the inner path of *Dynamic* is generally overapproximated, because the inner line segments are generated from the same parameter as the outer ones, the total point count is still about 15 percent lower than *Incremental*. *Hain* has suspiciously low timings and point counts, especially in the cubic case. Investigation shows that the processing of curves with sharp turns breaks early (see section 6.3). This invalidates the data so that we cannot make a meaningful assertion for these methods. AGG massively overapproximates the offset path, resulting in bad timings and point counts. So, once more, the incremental approach is generally faster than the recursive one while safely retaining a specified error bound, yet not strictly for rational curves. Again, there could be potential performance improvements by utilizing vectorization (except for the iterative parts) as well as general multi-threading.

6.3 Offset defects

Hain's offsetting methods are problematic for curves where cusps are appearing in the inner offset path. This usually happens for curves with sharp turns and will result in defective visual representations, as we may observe in figure 6.1.

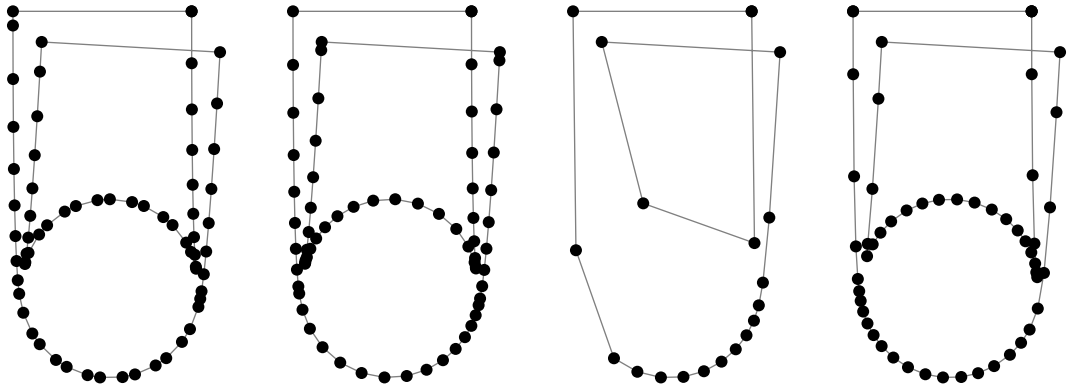

Figure 6.1: Offset path from *Incremental*, *Dynamic*, *Hain* and *AGG*

Although this behavior was mentioned by Hain in his paper [8], he did not explicitly provide a solution on how to prevent it with a reasonable performance impact.
The dynamic method exposes a similar behavior for extremely sharp turns (at least sharper than in figure 6.1) where the line segment skips the tip of the curve, but generally does not skip as far as *Hain*. The overall extent of the defect depends on the error bound and lessens for smaller values.

Chapter 7

Conclusion

Several algorithms to flatten and offset quadratic, cubic and rational curves within a specified error limit have been introduced. Cubic and rational curves are simplified to quadratic ones so that these can be either flattened or offset with efficient approaches for quadratic cases. The incremental methods for quadratic and cubic curves satisfy given error bounds while rational cases could produce some outliers. However, the approximation of rational curves is sufficient and even provides an option to reduce the outliers. Overall, the presented approaches achieve a better run-time performance than the other methods but tend to produce more line segments.

7.1 Future work

Although only labeled experimental, the dynamic approach for offsetting curves has demonstrated the benefit of alternative methods that process curves without the extra simplification step. It is almost twice as fast as the incremental approach while also generating about 15 percent less line segments. The drawback is the uncertain error bound which might be improved by predicting the course of the curvature radius. Furthermore, we have discussed two unfinished methods for flattening that could as well lead to faster solutions for offsetting in the long run.
Because of uniformly spaced curve segments, the incremental methods might benefit greatly from vectorization and multi-threading. Despite not being attempted, it would make a good starting point for extensive performance optimizations.

Bibliography

[1] Sheue-Ling Lien, Michael Shantz, Vaughan Pratt. Adaptive forward differencing for rendering curves and surfaces. *ACM Siggraph Computer Graphics 21, Issue 4*, pages 111–118, 1987.

[2] Gerald Farin. Algorithms for rational Bézier curves. *Computer-Aided Design 15, Issue 2*, pages 73–77, 1983.

[3] M. S. Floater. High order approximation of conic sections by quadratic splines. *Computer Aided Geometric Design 12, Issue 6*, pages 617–637, 1995.

[4] Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide (Fifth Edition)*. Morgan Kaufmann, 2002.

[5] Thomas F. Hain. Rapid Termination Evaluation for Recursive Subdivision of Bezier Curves. *Proceedings of the International Conference on Imaging Science, Systems and Technology*, pages 323–328, 2002.

[6] Athar L. Ahmad. Approximation of a Bézier Curve with a Minimal Number of Line Segments. Master's thesis, University of South Alabama, 2001.

[7] Thomas F. Hain, Athar L. Ahmad, David D. Langan. Precise Flattening of Cubic Bézier Segments. *Canadian Conference on Computational Geometry*, pages 180–183, 2004.

[8] Thomas F. Hain, Sri Venkat R. Racherla, David D. Langan. Fast, Precise Flattening of Cubic Bezier Segment Offset Curves. *17th Brazilian Symposium on Computer Graphics and Image Processing*, pages 244–249, 2004.

[9] Matthias Eck. Degree reduction of Bézier curves. *Computer Aided Geometric Design 10, Issues 3-4*, pages 237–251, 1993.

[10] Matthias Eck. Least squares degree reduction of Bézier curves. *Computer-Aided Design 27, Issue 11*, pages 845–851, 1995.

[11] Lizheng Lu, Guozhao Wang. Optimal multi-degree reduction of Bézier curves with G^2-continuity. *Computer Aided Geometric Design 23, Issue 9*, pages 673–683, 2006.

[12] Gershon Elber, In-Kwon Lee, Myung-Soo Kim. Comparing Offset Curve Approximation Methods. *IEEE Computer Graphics and Applications 17, Issue 3*, pages 62–71, 1997.

[13] Fabian Yzerman. Precise offsetting of quadratic Bézier curves, 2019.

[14] W3C. Scalable Vector Graphics (SVG) 2 W3C Candidate Recommendation. `https://www.w3.org/TR/SVG2/`, 2018.

[15] Rida T. Farouki. The Bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design 29, Issue 6*, pages 379–419, 2012.

[16] J. Sánchez-Reyes. Geometric recipes for constructing Bézier conics of given centre or focus. *Computer Aided Geometric Design 21, Issue 2*, pages 111–116, 2004.

[17] Qianqian Hu, Guojin Wang. Geometric meanings of the parameters on rational conic segments. *Science in China Series A: Mathematics 48, Issue 9*, pages 1209–1222, 2005.

[18] A. Cantón, L. Fernández-Jambrina, E. Rosado María. Geometric characteristics of conics in Bézier form. *Computer-Aided Design 43, Issue 11*, pages 1413–1421, 2011.

[19] Pieter J. Barendrecht. A gentle introduction to rational Bézier curves and NURBS, 2012.

[20] Rida T. Farouki. *Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable*. Springer, 2008.

[21] Dinesh Manocha, John F. Canny. Detecting cusps and inflection points in curves. *Computer Aided Geometric Design 9, Issue 1*, pages 1–24, 1992.

[22] D. J. Walton, D. S. Meek. Curvature extrema of planar parametric polynomial cubic curves. *Journal of Computational and Applied Mathematics 134*, pages 69–83, 2001.

[23] Maxim Shemanarev. Adaptive Subdivision of Bezier Curves. `http://antigrain.com/research/adaptive_bezier/`, 2005.