Shape control tools for curve and surface design

Doctoral Dissertation submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Andriamahenina Ramanantoanina

under the supervision of
Prof. Kai Hormann

September 2025

Dissertation Committee

Prof. Kai Hormann	Università della Svizzera italiana, Switzerland
Prof. Piotr Didyk	Università della Svizzera italiana, Switzerland
Prof. Igor Pivkin	Università della Svizzera italiana, Switzerland
Prof. Paweł Woźny	University of Wrocław, Wrocław, Poland
Prof. Bert Jüttler	Johannes Kepler University, Linz, Austria

Dissertation accepted on September 2025

Research Advisor	PhD Program Director
Prof. Kai Hormann	**Prof. Walter Binder / Prof. Wolf Stefan**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Andriamahenina Ramanantoanina
Lugano, September 2025

Once you start seeing results,
the grind becomes addictive.

Someone – probably

Abstract

Bézier curves are widely used in geometric modeling, computer-aided design, and computer graphics due to their intuitive control point-based shape manipulation and stable evaluation via the de Casteljau algorithm. Rational Bézier curves extend their capabilities by incorporating weights, which enables the exact representation of conic sections. This thesis investigates advanced shape control mechanisms for rational Bézier curves, including control point-based modifications, weight-based deformations, and interpolation-driven modifications.

We show that rational Bézier functions can be written in barycentric form. Essentially, there exists a one-to-one correspondence between these two representations. The barycentric form facilitates additional editing operations, such as shape modification with multiple constraints through interpolation points, interpolation points sliding, and shape-preserving point insertion. We show that these can be achieved by simultaneously updating the parameters of the barycentric form, namely, the interpolation points, the nodes, and the weights. We also show that a single weight manipulation affects the local curvature.

Beyond shape control, we analyse the computational efficiency of rational Bézier curve evaluation methods. Although the de Casteljau algorithm offers numerical stability, its quadratic complexity makes it impractical for high-degree curves. We conduct a comprehensive comparison of alternative evaluation techniques, providing insight into their relative performance.

We continue the exploration by studying shape control techniques for periodic rational Bézier curves. Similar manipulation techniques transfer analogously from the classical rational Bézier curves. We show that we can also write them in barycentric form, and we explore the editing capabilities enabled by the latter. Due to the apparent deviation from the control polygon in high-degree periodic Bézier representations, we introduce a family of trigonometric tangent interpolating curves. These curves are constructed to closely follow the control polygon by forcing the tangents to be parallel to a control edge at specific parameters. The second variant in this family mimics uniform cubic B-spline behaviour. These trigonometric curves are so far only defined for an odd number of control

points, consistent with the dimension of the space of trigonometric polynomials. We explore the construction of closed curves for an even number of points.

The final part of this thesis focuses on rational Bézier triangles, which are the natural extension of rational Bézier curves. We also explore the use of Lagrange interpolation. It is crucial to be able to describe geometrically the configuration of nodes so that such interpolating forms exist. Usually, uniformly distributed nodes are used. The Lagrange interpolation also exists if the nodes coincide with the vertices of a triangular grid. We propose a configuration where the nodes lie in nested triangles. We show that this configuration provides enough flexibility for the placement of nodes for low-degree surfaces.

Acknowledgements

I would like to thank all those involved in the GRAPES project, which enabled me to embark on this wonderful journey.

To my Ph.D. supervisor Prof. Kai Hormann, I am deeply grateful for the privilege of being one of your apprentices. Your guidance, support, and the countless experiences you have shared with me, both inside and outside academia, have shaped me as a researcher and as a person.

Thanks to all those who were (permanently or temporarily) resident in D4.11. Especially Chiara, Qingjun, and Nicky — we started at approximately the same time, and we suffered and enjoyed this journey together. I cannot express how much I have appreciated that.

Misaotra an'i dada sy neny izay nahita fony aho kely hoe ity taranja ity no mety ho ahy. Misaotra koa an'ireo zoky sy zandry, ny zafikelin'i bebe Sazy rehetra, an'ilay neny vaovao sy ny vady aman-janany tamin'ny fitrotroana am-bavaka sy ny fanohanana ara-moraly. Ny fitiavana sy ny faharetanareo no nankahery ahy hatrany, ary tsy ho tanteraka ity dia ity raha tsy teo ianareo.

Rindra, my honorable wife, misaotra nitondra hery vao teny antsasa-dalana. Misaotra nanohana ahy sy nihaino ireo hevitra rehetra mibosesika ao an-dohako, na dia taranja tsy misy idiranao sy tsy nianaranao aza. Tsy niala ianao, fa nihaino foana.

Dia ho feheziko amin'ny teny roa:

Mankasitraka aho...

Contents

Chapter 1

Introduction

Bézier curves are fundamental tools in geometric modeling, computer graphics, and computer-aided design, as they inherit intuitive shape-editing capabilities using control points and can be evaluated efficiently in linear time. Bézier curves are designed to roughly follow the shape of their control polygons. This allows for straightforward transformations such as translation, rotation, and scaling, which justify their popularity in various applications. We gain more flexibility by associating weights with control points and using rational Bézier curves. The weights come as an additional degree of freedom that affects the tightness of the control polygon and the resulting curve. By increasing a weight, the curve is pulled projectively towards the control point, and decreasing a weight pushes it away. This allows for exact representations of conic sections. There are several ways to manipulate the weights. They can be individually manipulated by dragging a point on the curve. They can also be updated simultaneously by introducing some shape factors, geometrically represented by the Farin points and the shoulder points. For simplicity, rational Bézier curves can be reparametrised so that the first and last weights are equal to 1. Despite these intuitive shape controls, the control points and the weights are external to the curve. And there are divided opinions even in the very early use of Bézier curves in car manufacturing on what is the best way to manipulate curves. The control points can be manipulated individually or simultaneously to achieve a single-point interpolation by dragging a point on the curve. The latter is a more direct way to handle Bézier curves. Single-point interpolation can be achieved using the least-squares technique. This construction can be further extended to allow for the tweaking of the local curvature. The existing techniques for the rational Bézier and interpolation methods are discussed in detail in Chapter 2–3. Then we proceed to recall our several findings in Chapter 4–9. We enumerate the contributions in

each chapter.

We first recall the shape control tools induced by the barycentric form. Interpolation of multiple points can be achieved by expressing a rational Bézier curve in barycentric form. This introduces additional degrees of freedom through interpolation points, weights, and nodes. We explore the editing capabilities induced by the modification of any of these parameters. Mainly, we show that there is a one-to-one correspondence between rational Bézier curves and barycentric forms. This means that we can reposition the interpolation points to modify rational Bézier curves. We show that simultaneous updates on the nodes and weights can result in sliding an interpolation point along a given curve or inserting additional points without changing its shape. Furthermore, we also show that a single weight manipulation can be used to adjust the local curvature. We show that barycentric forms can be reparametrised and written in standard form as well. The content of this chapter has been published in [93].
Ramanantoanina, A., & Hormann, K. (2021). New shape control tools for rational Bézier curve design. Computer Aided Geometric Design, 88, 102003.

We proceed by analysing different techniques to evaluate rational Bézier curves. In addition to shape-editing capabilities, the efficient evaluation of Bézier curves is a crucial consideration, particularly in real-time applications where computational cost is a limiting factor. The *de Casteljau algorithm* is widely used to evaluate polynomial and rational Bézier curves due to its numerical stability and elegant recursive formulation. However, it has quadratic time complexity, which becomes impractical for high-degree curves. To address this limitation, we present a comparative study of state-of-the-art evaluation algorithms, highlighting their computational efficiency. Our analysis provides insights into selecting the most appropriate evaluation techniques for various geometric modeling applications. The content of this chapter was taken from [53]. My contributions are essentially the efficiency analysis.
Fuda, C., Ramanantoanina, A., & Hormann, K. (2024). A comprehensive comparison of algorithms for evaluating rational Bézier curves. Dolomites Research Notes on Approximation, 17(3), 56–79.

In Chapter 6, we recall the shape control tools induced by the barycentric form for the design of closed curves. To design closed free-form curves, multiple curves can be patched together. However, it can be challenging to place control

points and set weights to obtain a high order of continuity at the joints. For example, we need at least six points to reproduce a circle with a single rational Bézier curve. Hence, in the case where the highest smoothness is necessary, we can use trigonometric curves, as they are essentially rational splines with continuity C^∞ at the joints. This can be achieved through periodic rational Bézier curves. They are designed to mimic the intuitive shape controls offered by classical Bézier curves. Similarly to classical rational Bézier curves, a periodic Bézier curve can also be written in trigonometric barycentric form. We explore the editing possibilities induced by the trigonometric barycentric form. We develop a simple formula to update the nodes and the weights to produce a point-sliding effect, as well as to insert new points without altering the shape of the curve. We also work out the formula for the tangents and show that modifying a weight individually affects the curvature of the curve. The content of this chapter was taken from our published results in [94].
Ramanantoanina, A., & Hormann, K. (2023). Shape control tools for periodic Bézier curves. Computer Aided Geometric Design, 103, 102193.

In Chapter 7, we introduce a new type of trigonometric curve. We introduce two novel representations of trigonometric polynomial curves, inspired by the concept of polynomial Gauss–Legendre curves. For a high number of control points, akin to classical Bézier curves, periodic Bézier curves tend to veer away from the shape of their polygons due to the properties of the basis functions. These new representations overcome this by predetermining the tangents at specific parameters. The first variant generates curves that closely follow the control polygon. The second variant produces curves that exhibit properties similar to uniform cubic B-spline curves, providing a more natural and visually appealing alternative for closed-curve design. The content of this chapter was taken from a conference paper [95].
Ramanantoanina, A., & Hormann, K. (2024). Trigonometric tangent interpolating curves. In R. Chen, T. Ritschel, & E. Whiting (Eds.), Pacific Graphics Conference Papers and Posters. The Eurographics Association.

There exist several workflows for designing (closed) curves. One is by simply adding new control points gradually; the other is to insert basic shapes such as circles, sketch the general silhouette of the intended final curve, and gradually refine the curve to be able to add details. The latter process is usually achieved through *subdivision* in 2D/3D modeling and it consists of doubling the number of control points. This motivates the need for the ability to construct curves

given by an even number of points. Therefore, in Chapter 8, we explore similar constructions for periodic Bézier curves and trigonometric tangent interpolating curves given by an even number of points. We also discuss some issues that arise with using trigonometric interpolation.

Bézier curves extend naturally in the surface case through Bézier triangles. They inherit similar manipulation techniques and can also be evaluated efficiently. In the curve case, a curve can be written in an interpolating form as long as the nodes are pairwise distinct. The case is rather complicated for the surface case. In general, this is the case if and only if there is no common algebraic curve that contains the nodes. There exist several descriptions of particular configuration of nodes, so that a Lagrange interpolation is well-defined. It is important to be able to describe these configurations geometrically. The simplest of all being uniformly distributed nodes. A bit more general configuration is that the nodes coincide with the vertices of a triangular grid. In Chapter 9, we show that a Lagrange interpolation is also well defined if the nodes form a set of nested triangles $T_1, \dots, T_{\lfloor (n+3)/3 \rfloor}$, with $n+4-3k$ nodes on each side of T_k. We observe that this configuration is flexible enough for a low-degree $n = 2, 3, 4$ surface design. For $n = 4$, the configuration can be further relaxed to allow the degeneracy of the inner triangle. We propose some manipulation technique to handle this case that guarantees a smooth transition from the degenerate to the non-degenerate case.

Chapter 2

Rational Bézier techniques

In the context of design, the most common representation of curves and surfaces is by polynomial parametrisation. The advantage of using polynomial functions is that they are smooth, making them ideal for creating visually pleasing sketches. Furthermore, they are easy to evaluate, using *Horner's scheme* [33], since they only involve basic arithmetic operations such as addition, multiplication, and exponentiation. We start our analysis with polynomial curves.

The most common way of expressing a polynomial curve $p : \mathbb{R} \to \mathbb{R}^d$ in d dimensions, where d is usually 2 or 3, of degree n is to use the monomial basis $\{1, t, \ldots, t^n\}$ and to write p as

$$p(t) = p_0 + p_1 t + \cdots + p_n t^n \tag{2.1}$$

with $p_0, \ldots, p_n \in \mathbb{R}^d$. In the context of design, this formulation is not ideal. The shape of a control net does not intuitively relate to the shape of a curve. And a given curve does not inherit affine transformation by applying affine transformation on its control net. However, affine invariance is a crucial characteristic, [18], because it means that a curve is independent of the choice of the coordinate system.

To overcome these issues, the following adjustments are introduced. First, naturally, a curve must have a start and an endpoint. Therefore, the parameter domain is restricted to an interval $[a, b]$. For simplicity, the standard be $a = 0$ and $b = 1$. Second, the points p_0 and p_n are given geometric meaning by making them the start and endpoint of the curve, that is, $p(0) = p_0$ and $p(1) = p_n$. This leads to the standard way of constructing polynomial curves.

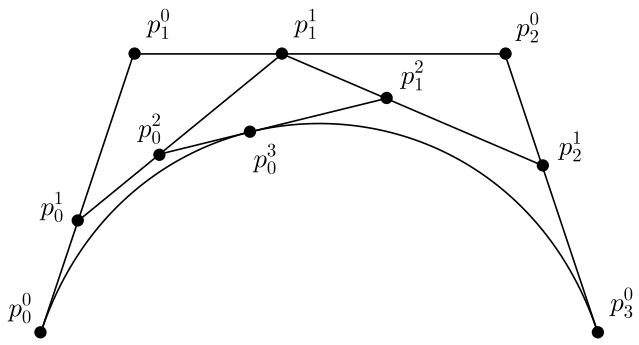


Figure 2.1. A geometric illustration of the de Casteljau algorithm that evaluates a cubic curve at $t = 0.4$.

2.1 Bézier curves

A curve given by only two points p_0 and p_1 is the line segment $[p_0, p_1]$. We note that each point q on this segment can be represented as a convex combination $q = p_0 s + t p_1$ where $s + t = 1$, which means $s = 1 - t$. We can then parametrise the line segment $[p_0, p_1]$ as *a convex combination*

$$p(t) = p_0(1-t) + t p_1, \qquad t \in [0,1]. \tag{2.2}$$

We can generalise this to construct polynomial curves of degree n by using a repetition of convex combinations. This is well known as the *de Casteljau algorithm* [20]

$$\begin{aligned} p_i^0(t) &= p_i, && i = 0,\dots,n \\ p_i^j(t) &= p_i^{j-1}(t)(1-t) + t p_{i+1}^{j-1}(t), && i = 0,\dots,n-j,\ j = 1,\dots,n. \end{aligned} \tag{2.3}$$

In this case, the curve is given by $p(t) = p_0^n$. It is clear that the algorithm (2.3) defines a polynomial curve of degree n since it involves n recursive linear interpolations. Such a curve is called *a Bézier curve* [17]. The points p_k are called *control points*, and together they form *a control net*. We cite some key properties of Bézier curves and, for each property, we state an example that highlights their importance.

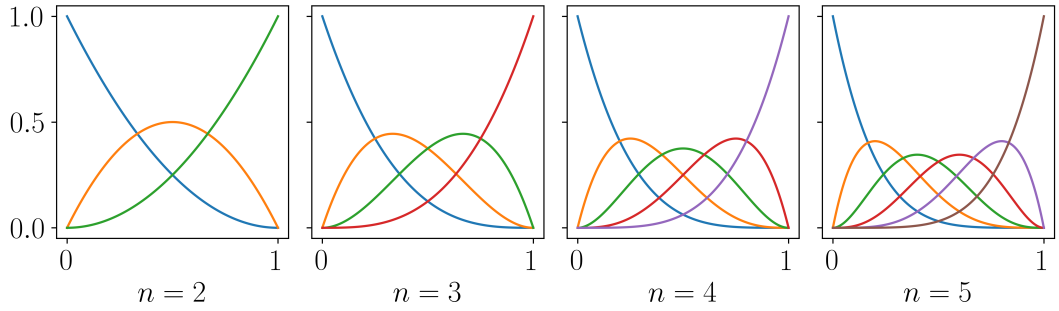


Figure 2.2. This shows the Bernstein polynomials (2.6) for $n = 2, 3, 4, 5$.

2.1.1 Bernstein–Bézier form

We can write a Bézier curve given by control points $p_0, \ldots, p_n \in \mathbb{R}^d$ as

$$\boldsymbol{b}(t) = \sum_{i=0}^{n} B_i^n(t) p_i, \qquad t \in [0,1] \tag{2.4}$$

such that, from the recursive algorithm (2.3), $B_k^n(t)$ is given by

$$\begin{aligned} &B_0^0(t) = 1, \\ &B_i^j(t) = B_i^{j-1}(t)t - (1-t)B_{i+1}^{j-1}(t), \quad i = 0, \ldots, n-j,\ j = 1, \ldots, n, \end{aligned} \tag{2.5}$$

where $B_k^n(t) = 0$ if $k < 0$ or $k > n$ [48]. It turns out that the polynomial $B_k^n(t)$ is a *Bernstein polynomial* [9], and can be written simply as

$$B_k^n(t) = \binom{n}{k}(1-t)^{n-k}t^k. \tag{2.6}$$

The properties of Bernstein polynomials justify the popularity of Bézier curves. We cite a few key properties that are relevant to curve design.

- **non-negativity**. For $t \in [0,1]$, $(1-t) \geq 0$. Therefore, $B_i^n(t) \geq 0$.
- **partition of unity**. $\sum_{i=0}^n B_i^n(t) = \sum_{i=0}^n \binom{n}{i}(1-t)^{n-i}t^i = (1-t+t)^n = 1$.
- **linear independence**. The Bernstein polynomial $B_k^n(t)$ can be written in monomial form as $B_k^n(t) = \sum_{i=k}^n (-1)^{i-k}\binom{n}{i}\binom{i}{k}t^i$. Hence, Bernstein polynomials can be obtained from the monomial basis under an invertible linear transformation, and they form a basis.
- **symmetry**. $B_k^n(t) = B_{n-k}^n(1-t)$. This induces the symmetry of control polygons to be reflected to the symmetry of curves.

From these constructions, we can extract a few key properties of the Bézier curves.

2.1.2 Affine invariance

A Bézier curve is affine invariant. In fact, we know that a line segment, which is a Bézier curve given by two points, is affine invariant. By induction on the number of points, we assume that all Bézier curves of degree with n control points are affine invariant. From the recursive formula (2.3), a curve with $n+1$ points is an affine combination of two degree $n-1$ Bézier curves. Therefore, Bézier curves are affine invariant.

2.1.3 Convex hull property

We can prove by induction in the same way as in 2.1.2 that a Bézier curve is contained in the convex hull of its control points. Alternatively, we can prove it by observing that for $t \in [0,1]$, the Bernstein polynomials are non-negative and that

$$\sum_{i=0}^{n} B_i^n(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i = ((1-t)+t)^n = 1 \tag{2.7}$$

that is, they form a partition of unity.

The convex hull property is very important and plays a key role, for example, in determining the intersections of two Bézier curves [109, 112].

2.1.4 Variation diminishing property

Bézier curves satisfy the *variation diminishing property* [2, 90]. That is, a curve $p(t)$ is cut by an arbitrary hyperplane no more often than its control polygon (see Figure 2.3). In fact, this is induced by the construction itself. Assume that a line L intersects a curve at $p(t_\star)$. Observing the illustration of the de Casteljau algorithm in Figure 2.1, a line that intersects the segment $[p_0^1(t_\star), p_1^1(t_\star)]$ must intersect either the segment $[p_0^1(t_\star), p_1^0(t_\star)]$ or the segment $[p_1^0(t_\star), p_1^1(t_\star)]$. Generally, a line that intersects the segment $[p_i^j(t_\star), p_{i+1}^j(t_\star)]$ must intersect either the segment $[p_i^j(t_\star), p_{i+1}^{j-1}(t_\star)]$ or the segment $[p_{i+1}^j(t_\star), p_{i+1}^{j-1}(t_\star)]$. However, in general, the converse is not true. From the assumption that L intersects the curve at $p(t_\star)$, this means that L intersects the segment $[p_0^{n-1}(t_\star), p_1^{n-1}(t_\star)]$. By iteration, L must intersect one of the control edges.

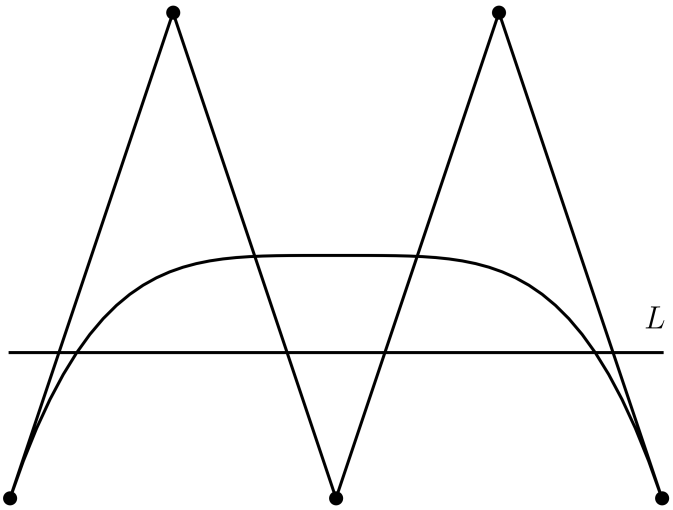


Figure 2.3. Observe that the line L intersects the control polygon four times while only intersecting the curve twice.

2.2 Rational Bézier curves

Polynomial curves and splines are essential tools for creating most free-form curves, some of the most popular methods described [28, 46, 127]. It is also important to be able to represent some "primary" curves such as conic arcs. The latter is usually represented in trigonometric form; however, it is well-known that trigonometric functions are simple rational functions.

$$\cos\theta = \frac{1-t^2}{1+t^2}, \qquad \sin\theta = \frac{2t}{1+t^2}, \qquad t = \tan\frac{\theta}{2}.$$

Hence, the Bézier curves are extended to the rational setting by introducing a set of *weights* $w_0, \ldots, w_n$. A rational Bézier curves is given by

$$p(t) = \frac{\sum_{i=0}^{n} B_i^n(t) w_i p_i}{\sum_{i=0}^{n} B_i^n(t) w_i}. \tag{2.8}$$

Now, a conic arc can be represented by quadratic rational Bézier curves (see Figure 2.10). For design purposes, the weights are kept positive in order to obtain continuous curves. Note that by the partition of unity property, having equal weights results in a polynomial Bézier curve.

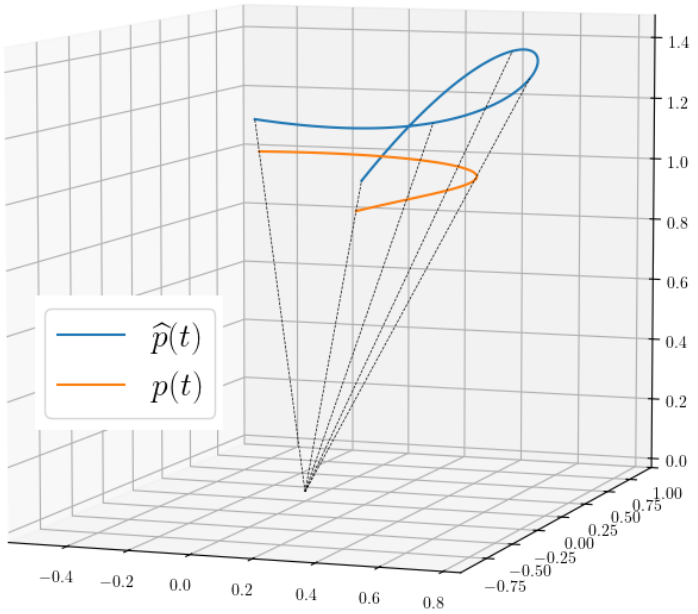


Figure 2.4. This illustrates the fact that a rational Bézier curve is a projection onto the plane $z = 1$ of a spatial polynomial Bézier curve.

2.2.1 Projection of spatial curves

Rational Bézier curves can be considered as projections of polynomial Bézier curves in higher dimensions (see Figure 2.4). In fact, let $\widehat{p}_i = (w_i p_i, w_i)$ for $i = 0, \dots, n$ and

$$\widehat{p}(t) = \sum_{i=0}^{n} B_i^n(t) \widehat{p}_i. \tag{2.9}$$

The rational Bézier curve (2.8) is the image of the curve $\widehat{p}(t)$ in (2.9) under the projection

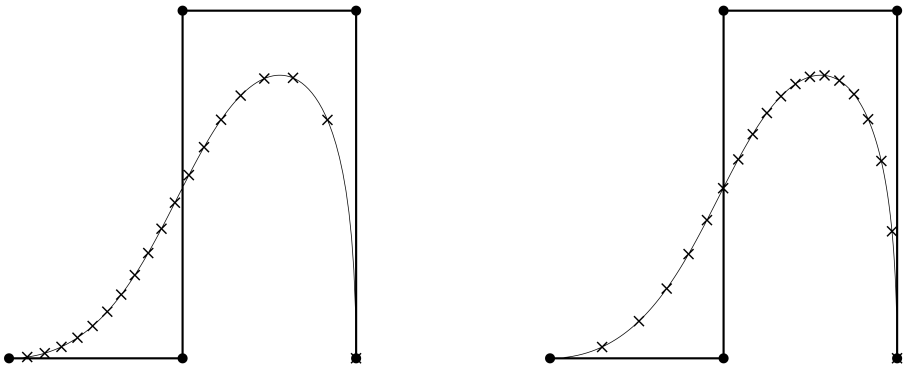
$$\Pi : (\widehat{x}, \widehat{y}, \widehat{z}) \mapsto (x, y) = (\widehat{x}/\widehat{z}, \widehat{y}/\widehat{z}). \tag{2.10}$$

2.2.2 Standard form

As Bézier curves are invariant under linear reparametrisation, rational Bézier curves are invariant under rational linear (Möbius) transformations [48]

$$t \mapsto \frac{(1-\alpha)t}{\alpha(1-t) + (1-\alpha)t}, \qquad \alpha \in (0, 1). \tag{2.11}$$

Transformations (2.11) preserve the control polygon and the shape of the curve but not the weights (see Figure 2.5). Hence, it is natural to ask when two sets of



(a) (b)

Figure 2.5. This illustrates the difference between two parametrisations of the same curve (2.8) with weights $w_k = (n+1-k)^4$ (a) and $w_0 = w_4 = 1$, $w_1 \approx 1.71$, $w_2 \approx 2.41$, $w_3 \approx 2.4$ (b). The crosses visualise the curve points $p(k/20)$, $k = 1, \ldots, 19$.

weights $w_0, \ldots, w_n$ and $v_0, \ldots, v_n$ represent the same shape. As a matter of fact, Patterson [82] shows that it is true if there exists some non-zero scalar λ such that

$$v_i = \lambda^{n-i} w_i, \qquad i = 0, \ldots, n. \tag{2.12}$$

Consequently, a rational Bézier curve can be written in *standard form* with weights $v_0, \ldots, v_n$ such that $v_0 = v_n = 1$, [47], by taking

$$\lambda = \left(\frac{w_n}{w_0}\right)^{1/n}. \tag{2.13}$$

2.2.3 Rational de Casteljau algorithm

To evaluate a rational Bézier curve, a simple approach would be to evaluate $\widehat{p}$ and project the final result onto the plane $z = 1$. Alternatively, Farin [43] extends the de Casteljau algorithm (2.3) to the rational setting.

$$\begin{array}{ll} p_i^0 = p_i, & i = 0, \ldots, n, \\ w_i^0 = w_i, & i = 0, \ldots, n, \\ w_i^j = w_i^{j-1}(1-t) + t w_{i+1}^{j-1}, & i = 0, \ldots, n-j, j = 1, \ldots, n, \\ p_i^j = \frac{w_i^{j-1} p_i^{j-1}(1-t) + t w_{i+1}^{j-1} p_{i+1}^{j-1}}{w_i^j}, & i = 0, \ldots, n-j, j = 1, \ldots, n. \end{array} \tag{2.14}$$

2.2.4 Differentiation of rational Bézier expressions

We first look at the differentiation of Bernstein polynomials. They are given by

$$\frac{dB_k^n(t)}{dt} = n\left[B_{k-1}^{n-1}(t) - B_k^{n-1}(t)\right]. \tag{2.15}$$

To that derivatives of Bézier curves (2.4) are given by [60]

$$p'(t) = n\sum_{i=0}^{n-1} B_i^{n-1}(t)\Delta_i, \qquad \Delta_i = p_{i+1} - p_i. \tag{2.16}$$

Now we can derive the differentiation of rational Bézier curves by writing (2.8) as $p(t) = N(t)/D(t)$, where $N(t)$ and $D(t)$ are polynomial Bézier curves. By noting that $D(t)p(t) = N(t)$, it is easy to see that the k-th derivative of p is given by

$$p^{(k)}(t) = \frac{N^{(k)}(t) - \sum_{i=1}^{k}\binom{k}{i}D^i(t)p^{(k-i)}(t)}{D(t)}. \tag{2.17}$$

2.3 Shape control for rational Bézier curves

We study the modification of the degree-of-freedoms, namely, the control points and the weights. They can be manipulated individually or simultaneously to achieve the desired manipulation effect.

2.3.1 Single control point repositioning

Moving a point from its current position p_k to a new position $\widetilde{p}_k$ means that we add the normalised basis function, scaled by $\Delta_k = \widetilde{p}_k - p_k$, to p to get the new curve

$$\widetilde{p} = p + \frac{B_k^n w_k}{\sum_{i=0}^{n} B_i^n w_i}\Delta_k.$$

If the weights are all equal, the Bernstein polynomial $B_k^n(t)$ is maximal at $t = k/n$, hence $p(k/n)$ is the point most affected by this manipulation.

2.3.2 Least-squares technique for single point interpolation

Consider a point $q = p(t_\star)$ on the curve for some $t_\star \in (0,1)$. Suppose that we move it to a new position $\widetilde{q}$ (see Figure 2.6). Assume that we want to reposition

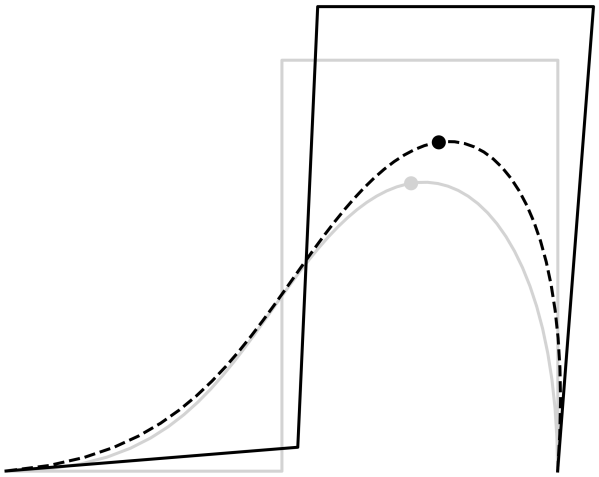


Figure 2.6. This illustrates the manipulation techniques using least-squares for a single point interpolation.

the points $p_k, \ldots, p_{k+l}$ in new positions $\widetilde{p}_k, \ldots, \widetilde{p}_{k+l}$ such that $\widetilde{q} = \widetilde{\boldsymbol{p}}\boldsymbol{b}(t_\star)^T$ where $\widetilde{\boldsymbol{p}} = (\widetilde{p}_0, \ldots, \widetilde{p}_n)$ and

$$\boldsymbol{b}(t) = \frac{1}{\sum_{i=0}^{n} B_i^n(t) w_i}(0, , \ldots, 0, B_k^n(t)w_1, \ldots, B_{k+l}^n(t)w_{n-1}, 0, \ldots, 0)^T.$$

Note that the endpoints are preserved. We can use the method of Lagrange multipliers [23] to minimise $\sum_{i=0}^{n} \|\widetilde{p}_i - p_i\|^2$. A technique in [8, 45] shows that we can determine the solution of this least square problem by writing the solution $\widetilde{\boldsymbol{p}}$ in the form

$$\widetilde{\boldsymbol{p}} = \boldsymbol{p} + \lambda \boldsymbol{b}^T(t_\star), \tag{2.18}$$

for some $\lambda \in \mathbb{R}^2$ and by assuming $\widetilde{p}_k$, p_k, for $k = 0, \ldots, n$, $\widetilde{q}$, q, and λ as column vectors. By multiplying to the right by $\boldsymbol{b}(t_\star)$ on each term, it follows that

$$\widetilde{q} = q + \lambda \boldsymbol{b}^T(t_\star)\boldsymbol{b}(t_\star).$$

Finally, we obtain the value of λ in (2.18) as

$$\lambda = \frac{\widetilde{q} - q}{\boldsymbol{b}^T(t_\star)\boldsymbol{b}(t_\star)}. \tag{2.19}$$

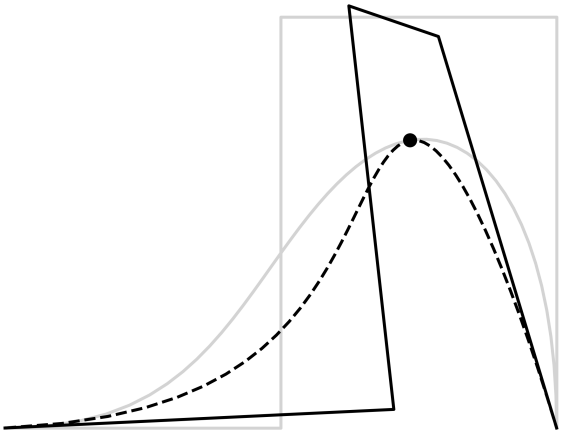


Figure 2.7. This illustrates the manipulation techniques using least-squares for a tangent interpolation.

2.3.3 Least-squares technique for tangent interpolation

We further extend this method to devise a formula that determines the control points while changing the tangent at q. Assume $p'(t_\star) = q'$ and we want to tweak it to a new value $\widetilde{q}'$. Using the same approach, we write the solution in the form

$$\widetilde{\boldsymbol{p}} = \boldsymbol{p} + \lambda_1 \boldsymbol{b}^T(t_\star) + \lambda_2 \boldsymbol{b}'^T(t_\star), \tag{2.20}$$

for some column vectors $\lambda_1, \lambda_2 \in \mathbb{R}^2$. By right multiplication by $\boldsymbol{b}(t_\star)$, we have

$$\widetilde{q} = q + \lambda_1 \boldsymbol{b}^T(t_\star)\boldsymbol{b}(t_\star) + \lambda_2 \boldsymbol{b}'^T(t_\star)\boldsymbol{b}(t_\star). \tag{2.21}$$

Similarly, by right multiplication by $\boldsymbol{b}'(t_\star)$, we have

$$\widetilde{q}' = q' + \lambda_1 \boldsymbol{b}^T(t_\star)\boldsymbol{b}'(t_\star) + \lambda_2 \boldsymbol{b}'^T(t_\star)\boldsymbol{b}'(t_\star). \tag{2.22}$$

By solving the system of equations (2.21) and (2.22), we find the values of the parameters λ_1 and λ_2 in (2.20) as

$$\begin{aligned} \lambda_1 &= -\lambda_2 \frac{\boldsymbol{b}'^T(t_\star)\boldsymbol{b}(t_\star)}{\|\boldsymbol{b}(t_\star)\|^2} \\ \lambda_2 &= (\widetilde{q}' - q') \frac{\|\boldsymbol{b}(t_\star)\|^2}{(\boldsymbol{b}^T(t_\star)\boldsymbol{b}'(t_\star))^2 + \|\boldsymbol{b}(t_\star)\|^2 \|\boldsymbol{b}'(t_\star)\|^2} \end{aligned} \tag{2.23}$$

Note that we can also change the curvature at a point by simply scaling the strength of the tangent vector as we see in Figure 2.7.

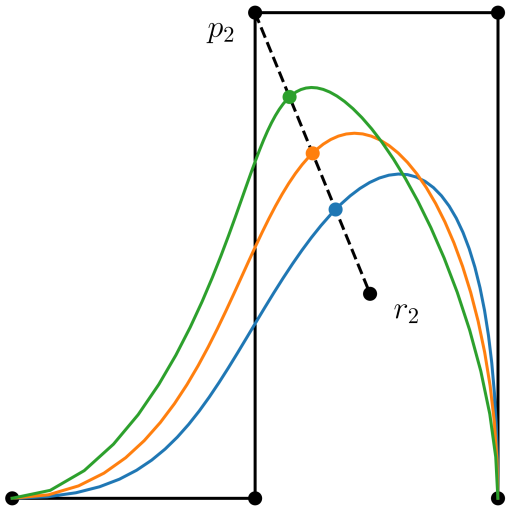


Figure 2.8. This shows the effect of changing the weight w_2 individually by dragging $q_\star$ along the segment $[p_2, r_2]$.

2.3.4 Changing a weight individually via auxiliary points

Weights are used to push or pull a curve towards its control polygon [84, 86]. We show that individually manipulating the weights can be done by directly dragging a point on the curve with the help of an auxiliary point (see Figure 2.8).

Proposition 2.1. *Let $t_\star \in (0,1)$, $q = p(t_\star)$, and we want to move q to a new position $\widetilde{q} = \widetilde{p}(t_\star)$. Let $r_k = p(t_\star)$ assuming $w_k = 0$. For some scalars $\alpha, \widetilde{\alpha}$ such that*

$$\begin{cases} q = (1-\alpha) r_k + \alpha p_k, \\ \widetilde{q} = (1-\widetilde{\alpha}) r_k + \widetilde{\alpha} p_k. \end{cases}$$

Then the new weight of p_k is given by

$$\widetilde{w}_k = \frac{\widetilde{\alpha}}{1-\widetilde{\alpha}} \frac{1-\alpha}{\alpha} w_k. \tag{2.24}$$

Proof. From the relation, $q = p(t_\star)$, we have

$$(1-\alpha) r_k + \alpha p_k = \frac{\sum_{i=0}^n B_i^n(t_\star) w_i p_i}{\sum_{i=0}^n B_i^n(t_\star) w_i} = \frac{r_k + B_k w_k p_k}{1 + B_k w_k},$$

where $B_k = B_k^n(t_\star) / \sum_{i=0, i\neq k}^n B_i^n(t_\star) w_i$. This means that

$$\alpha = \frac{B_k w_k}{1 + B_k w_k},$$

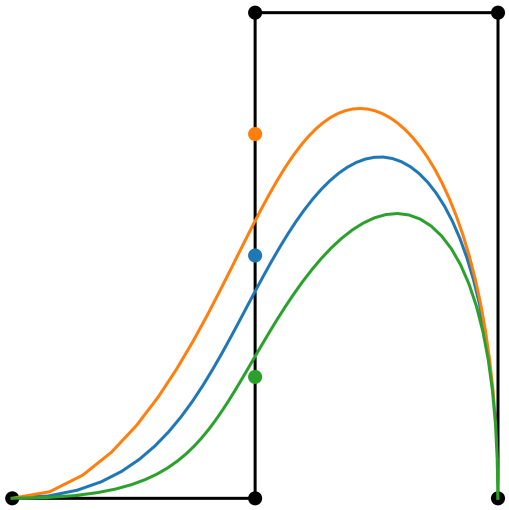


Figure 2.9. This illustrates the effect of dragging the Farin point F_1 along the edge $[p_1, p_2]$.

which is equivalent to writing

$$w_k = \frac{\alpha}{1-\alpha}\frac{1}{B_k}. \tag{2.25}$$

By the same manner, one can show that

$$\widetilde{w}_k = \frac{\widetilde{\alpha}}{1-\widetilde{\alpha}}\frac{1}{B_k}. \tag{2.26}$$

The proposition follow easily from (2.25) and (2.26). □

Note that we can easily compute α and $\widetilde{\alpha}$ respectively as

$$\alpha = \frac{\|q - r_k\|}{\|p_k - r_k\|}, \qquad \widetilde{\alpha} = \frac{\|\widetilde{q} - r_k\|}{\|p_k - r_k\|}.$$

2.3.5 Changing a weight via Farin points

We can also manipulate the weights simultaneously with the help of shape factors. The first shape factors that we mention are related to the *Farin points* F_i, $i = 0, \ldots, n-1$ [43] (see Figure 2.9). They are weighted average of two consecutive control points, that is,

$$F_i = \frac{P_i w_i + w_{i+1} P_{i+1}}{w_i + w_{i+1}}. \tag{2.27}$$

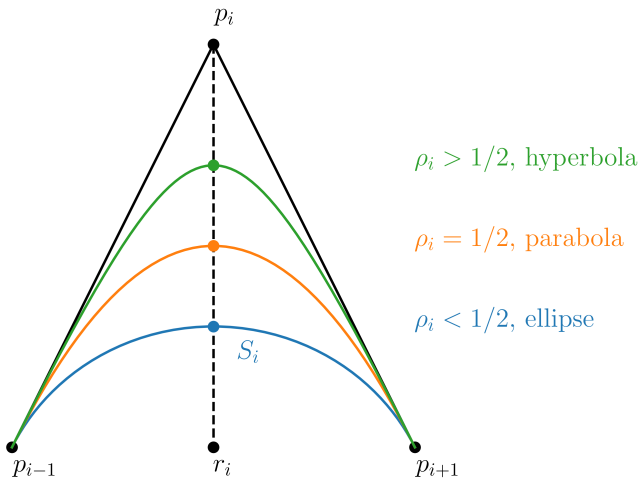


Figure 2.10. Effect of varying the shape factor ρ_i on the conic c_i.

These points are determined by a shape factor σ_i such that

$$\sigma_i = \frac{w_{i+1}}{w_i}, \qquad i = 0, \dots, n-1. \tag{2.28}$$

From (2.27) and (2.28), we deduce that

$$F_i = \frac{P_i + \sigma_i P_{i+1}}{1 + \sigma_i}.$$

Hence by dragging a Farin point F_k along the edge $[p_k, p_{k+1}]$, from (2.28), we can update the weights as

$$\begin{cases} w_{k+1} = w_k \frac{\|p_k - F_k\|}{\|p_{k+1} - F_k\|}, \\ w_{i+1} = w_i \sigma_i, \text{ for } i = k+1, \dots, n-1. \end{cases} \tag{2.29}$$

We recall that we can write (2.8) in standard form by taking weights $v_i = \lambda^{n-i} w_i$, for λ as in (2.13). This implies that σ_i is scaled by $1/\lambda$. Hence, the Farin points are not invariant under a Möbius transformation.

2.3.6 Changing a weight via shoulder points

We recall another kind of shape factor that is derived from conics, namely, the *shoulder point* [43] (cf. Figure 2.11). In this case, σ_i is given by

$$\sigma_i = \frac{w_i^2}{w_{i-1} w_{i+1}}, \qquad i = 1, \dots, n-1. \tag{2.30}$$

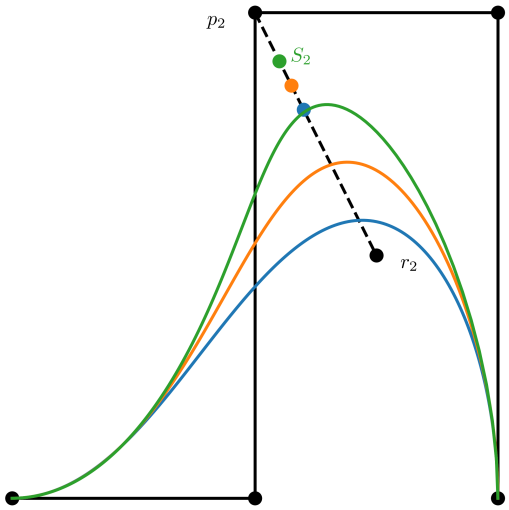


Figure 2.11. Effect of sliding the shoulder point S_2 on the segment $[p_2, r_2]$.

It is clear that σ_i is now invariant under Möbius transformation (2.12) [106]. The factor σ_i determines the conic c_i (2.8) defined by p_{i-1}, p_i, p_{i+1} and w_{i-1}, w_i, w_{i+1} [72, 121]. As σ_i can grow indefinitely, for design purposes, it is preferable to use a factor $\rho_i \in (0,1)$ [76, 115, 116] that determines the position of a shoulder point $S_i = (1-\rho_i)r_i + \rho_i p_i$, for $r_i = (p_{i-1} + p_{i+1})/2$ (see Figure 2.10). The shape factor σ_i and ρ_i are related as [1, 124]

$$\rho_i = \frac{\sqrt{\sigma_i}}{1+\sqrt{\sigma_i}}. \tag{2.31}$$

Therefore, the factor ρ_i also determines the conic c_i.

Now in order to update the weights after adjusting the shape factor ρ_k, first we update σ_k as

$$\sigma_k = \frac{\rho_k^2}{(1-\rho_k)^2},$$

and we can update the weights as

$$w_{i+1} = \frac{w_i^2}{w_{i-1}\sigma_i}, \qquad i = k, \ldots, n-1.$$

Note that a curve p is a blend of the conics c_i as each of them is traced by an intermediate point $c_i(t) = p_{i-1}^2(t)$ on the second level of the rational de Casteljau algorithm (2.14) (see Figure 2.12). This induces a more pseudo-local manipulation than the manipulations described in 2.3.4 and 2.3.5.

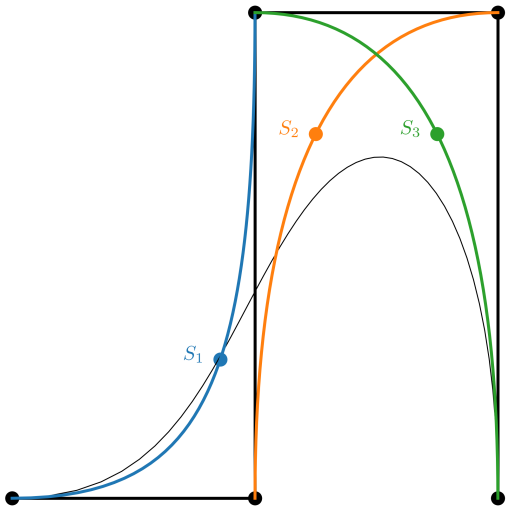


Figure 2.12. This illustrates the conics c_i which are obtained at intermediate steps of the rational de Casteljau algorithms.

2.3.7 Degree elevation

In addition to manipulating existing control points and weights, it is possible to increase the number of points and weights (see Figure 2.13). This process is called *degree elevation* [43, 60]. Let $\widetilde{p}_0,\ldots,\widetilde{p}_{n+1}$, $\widetilde{w}_0,\ldots,\widetilde{w}_{n+1}$ define the degree elevated rational curve (2.8). Considering the notations used for the rational de Casteljau algorithm (2.14), we have

$$\begin{cases} \widetilde{p}_i = p_{i-1}^1\left(\frac{n+1-i}{n+1}\right), \\ \widetilde{w}_i = w_{i-1}^1\left(\frac{n+1-i}{n+1}\right), \end{cases} \tag{2.32}$$

for $i = 0,\ldots,n+1$. By progressively raising the degree, the control polygon converges to the curve and can be used to compute its length [96].

2.3.8 Degree reduction

Degree reduction is a reverse process for degree elevation [25, 41, 51]. It is useful if the number of points is more than necessary. For the sake of simplicity, we write the process in a homogeneous form. Let $\widetilde{\boldsymbol{p}} = \left(\binom{\widetilde{w}_0\widetilde{p}_0}{\widetilde{w}_0},\ldots,\binom{\widetilde{w}_{n-1}\widetilde{p}_{n-1}}{\widetilde{w}_{n-1}}\right)^T$ and $\widehat{\boldsymbol{p}} = (\widehat{p}_0,\ldots,\widehat{p}_n)^T$. Similar relations to (2.32) can be written as

$$\widehat{p}_i = \frac{1}{n}\left(\binom{\widetilde{w}_{i-1}\widetilde{p}_{i-1}}{\widetilde{w}_{i-1}} i + (n-i)\binom{\widetilde{w}_i\widetilde{p}_i}{\widetilde{w}_i}\right).$$

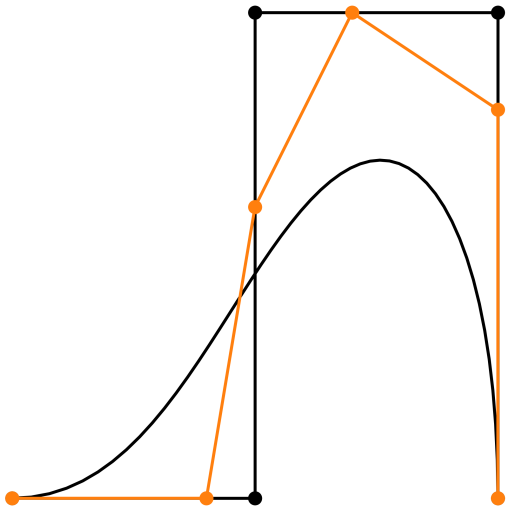


Figure 2.13. This illustrates the control polygon after applying one degree elevation.

Then we have $\widehat{\boldsymbol{p}} = M\widetilde{\boldsymbol{p}}$ where

$$M = \left(m_{ij}\right)_{0\le i\le n,\ 0\le j\le n-1}, \qquad m_{ij} = \begin{cases} \frac{n-i}{n}, & \text{if } i = j, \\ \frac{i}{n}, & \text{if } i = j+1, \\ 0, & \text{otherwise.} \end{cases}$$

The best degree reduced solution [45] is given by the Moore-Penrose inverse [69]

$$\widetilde{\boldsymbol{p}} = (M^T M)^{-1} M^T \widehat{\boldsymbol{p}}.$$

For design purposes, it is customary to force the endpoints to be equal (see Figure 2.14).

2.3.9 Rational Bézier splitting

During a design process, the designer could be satisfied with a part of the curve and want to maintain it unaffected by further manipulations. This can be done by splitting the curves into two parts at any $u \in (0,1)$. From the rational de Casteljau algorithm (2.14), a curve b can be split into two parts [7]

- b_0 defined by p_0^i with the respective weight w_0^i for $i = 0,\dots,n$,
- b_1 defined by p_i^{n-i} with the respective weight w_i^{n-i} for $i = 0,\dots,n$.

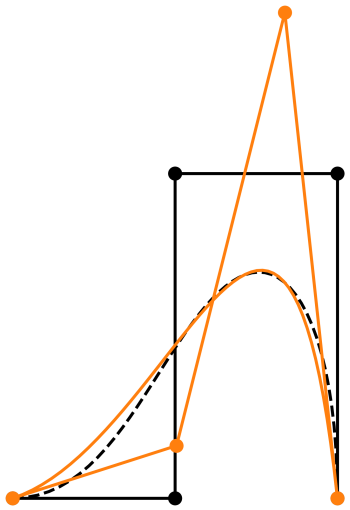


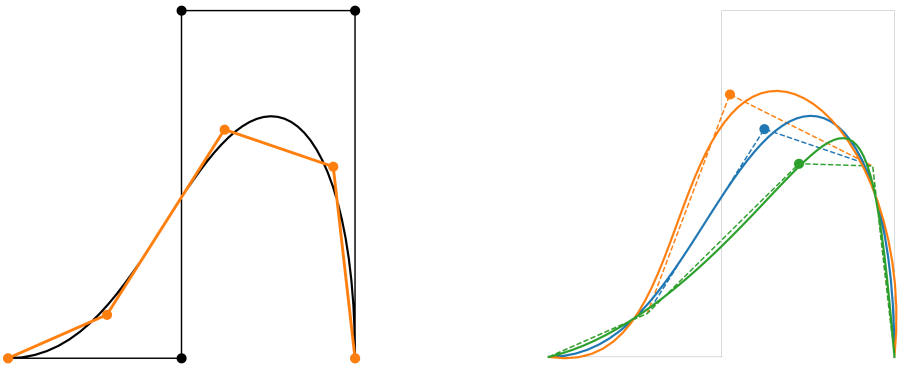
Figure 2.14. In general, a degree reduction is not possible. This shows the approximation of our initial curve with a lower degree curve.

2.3.10 Manipulation with different polygons

There exists a diverse type of control polygon that one can use to manipulate rational Bézier curves. These control polygons are associated with other types of curves, such as rational Wang-Ball curves [37, 83, 120], rational DP curves [35, 38, 39], and other similar types of curves [36]. In this thesis, we are going to study two constructions that have control polygons that are tighter to the actual curve than the Bézier control polygons. The first is the control polygon where the vertices are interpolated. This will be the focus of the next chapter. The second type produces the so-called Gauss–Legendre curves [77]. To the best of our knowledge, they are only defined for polynomial curves. And in contrast to Bézier curves, they do not satisfy the end-tangent property (see Figure2.15). To remedy this, one can use the Gauss–Lobatto control polygon [71].

Let $q_0, \ldots, q_n$ be the control points that form the Gauss–Legendre control polygon. Let $\tau_0, \ldots, \tau_{n-1}$ be the roots of the degree n Legendre polynomial. Then the Gauss–Legendre control polygon satisfies

$$p'(\tau_i) = \frac{q_{i+1} - q_i}{\omega_i}, \qquad i = 0, \ldots, n-1, \tag{2.33}$$



(a) (b)

Figure 2.15. (a) displays the Gauss–Legendre control polygon of a Bézier curve. (b) demonstrates the effectiveness of the manipulation with Gauss–Legendre polygon.

where the scalar ω_i are closely related to the Lagrange polynomials as

$$\omega_i = \int_{-1}^{1} \prod_{j=0, j\neq i}^{n-1} \frac{t-t_j}{t_i-t_j} dt.$$

Since the end-points are the same, we can obtain the Gauss–Legendre control polygon from the Bézier control polygon as

$$\begin{cases} q_0 = p_0, \\ q_i = q_{i-1} + \omega_{i-1} p'\left(\frac{\tau_{i-1}+1}{2}\right), \ i = 1, \ldots, n. \end{cases}$$

Conversely, let $\Delta_i^p = n(p_{i+1} - p_i)$ and $\Delta_i^q = (q_{i+1} - q_i)/\omega_i$. The relations (2.33) can be written compactly as $M\Delta_p = \Delta_q$ where $\Delta_p = (\Delta_0^p, \ldots, \Delta_{n-1}^p)^T$, $\Delta_q = (\Delta_0^q, \ldots, \Delta_{n-1}^q)^T$, and

$$M = \left(B_j^{n-1}\left(\frac{\tau_i + 1}{2}\right)\right)_{i,j\in[0,\ldots,n-1]}$$

We can extract $\Delta_p = M^{-1}\Delta_q$, and we have the Bézier control points as

$$\begin{cases} p_0 = q_0, \\ p_i = p_{i-1} + \frac{\Delta_i^p}{n}, i = 1, \ldots, n. \end{cases}$$

2.4 Evaluation techniques

For clarity of the notation, we use capital letters P_i, R_i, S_i for points and w_i, u_i for weights. We analyse the existing evaluation algorithms of the rational Bézier curve

$$P(t) = \frac{\sum_{i=0}^{n} B_i^n(t) w_i P_i}{\sum_{i=0}^{n} B_i^n(t) w_i}. \tag{2.34}$$

2.4.1 Rational de Casteljau algorithms

The most straightforward approach to compute $P(t)$ is by using the classic quadratic time de Casteljau algorithm for polynomials as in (2.3) [20]. In the case of a rational Bézier curve of the type in (2.34), we recall that it can be considered as the central projection of the spatial polynomial curve

$$\widehat{P}(t) = \sum_{i=0}^{n} B_i^n(t) \widehat{P}_i, \qquad \widehat{P}_i = \begin{pmatrix} w_i P_i \\ w_i \end{pmatrix}, \tag{2.35}$$

under the projection (2.10). This implies that we can apply the classical de Casteljau algorithm to $\widehat{P}(t)$ and then project the final result according to (2.10) (Algorithm 2 and 3). This process is equivalent to first computing the values of the numerator $N(t)$ and the denominator $D(t)$ with the recursive formulas 2.3.

$$\begin{cases} N_i^0 = w_i P_i, \\ N_i^r = N_i^{r-1}(1-t) + N_{i+1}^{r-1} t, \end{cases} \quad \text{and} \quad \begin{cases} D_i^0 = w_i, \\ D_i^r = D_i^{r-1}(1-t) + D_{i+1}^{r-1} t, \end{cases} \tag{2.36}$$

$i = 0, \ldots, n$ and $r = 1, \ldots, n$, respectively, and then the final result as $P(t) = N_0^n / D_0^n$. We also note that this method exhibits quadratic complexity. Alternatively, Farin [43] adapts this approach into a more robust quadratic time algorithm (Algorithm 4) with additional geometric meaning in 2.14.

2.4.2 Horner-like algorithms

Schumaker and Volk [110] are the first to achieve an algorithm for computing polynomial Bézier curves with linear time complexity. Their idea is to use nested multiplications for the computation, which results in a significant gain in terms of efficiency. We present a straightforward extension of the VS algorithm by first applying it to the numerator and the denominator of $P(t)$, and then simplifying some common factors. In particular, we express the rational Bézier curve

in (2.34) equivalently as

$$P(t) = \frac{\sum_{i=0}^{n} x^{n-i}\binom{n}{i} w_i P_i}{\sum_{i=0}^{n} x^{n-i}\binom{n}{i} w_i}, \qquad x = \begin{cases} (1-t)/t, & t > 1/2 \\ t/(1-t), & t \le 1/2. \end{cases} \tag{2.37}$$

There are many methods for evaluating a polynomial; Goldman [57] highlights two approaches: one using a Horner scheme and the other employing a ladder pattern. However, Warren [123] shows that these forms are equivalent for the monomial basis, so we consider the former. Therefore, the VS algorithm evaluates the numerator and the denominator using a Horner scheme (Algorithm 5 and 6) as

$$P(t) = \begin{cases} \dfrac{\binom{n}{n} w_n P_n + x\Big(\binom{n}{n-1} w_{n-1} P_{n-1} + \cdots + x\Big(\binom{n}{1} w_1 P_1 + x\binom{n}{0} w_0 P_0\Big)\ldots\Big)}{\binom{n}{n} w_n + x\Big(\binom{n}{n-1} w_{n-1} + \cdots + x\Big(\binom{n}{1} w_1 + x\binom{n}{0} w_0\Big)\ldots\Big)}, & t > 1/2, \\ \dfrac{\binom{n}{0} w_0 P_0 + x\Big(\binom{n}{1} w_1 P_1 + \cdots + x\Big(\binom{n}{n-1} w_{n-1} P_{n-1} + x\binom{n}{n} w_n P_n\Big)\ldots\Big)}{\binom{n}{0} w_n + x\Big(\binom{n}{1} w_1 + \cdots + x\Big(\binom{n}{n-1} w_{n-1} + x\binom{n}{n} w_n\Big)\ldots\Big)}, & t \le 1/2. \end{cases}$$

With the same strategy, Farin [45] presents another Horner-like algorithm (Algorithm 5 and 7) by setting $s = 1 - t$ and computing $P(t)$ in (2.34) as

$$\begin{aligned} P(t) &= \frac{\sum_{i=0}^{n} t^i s^{n-i}\binom{n}{i} w_i P_i}{\sum_{i=0}^{n} t^i s^{n-i}\binom{n}{i} w_i} \\ &= \frac{\Big(\ldots\Big(\binom{n}{0} w_0 P_0 s + \binom{n}{1} w_1 P_1 t\Big)s + \cdots + \binom{n}{n-1} w_{n-1} P_{n-1} t^{n-1}\Big)s + \binom{n}{n} w_n P_n t^n}{\Big(\ldots\Big(\binom{n}{0} w_0 s + \binom{n}{1} w_1 t\Big)s + \cdots + \binom{n}{n-1} w_{n-1} t^{n-1}\Big)s + \binom{n}{n} w_n t^n}. \end{aligned} \tag{2.38}$$

2.4.3 Geometric approach

On the one hand, while the rational adaptation of Casteljau by Farin [43] has a nice geometric interpretation, it can only be done in quadratic time. On the other hand, the VS algorithm has linear time complexity, but it lacks geometric interpretation and properties. For this reason, Woźny and Chudy [125] introduces a new linear time algorithm that has a nice geometric interpretation (see Figure 2.16). In particular, $P(t)$ can be computed recursively (Algorithm 8) using a Horner-like scheme and convex combinations as

$$\begin{cases} h_0 = 1, & h_i = \dfrac{w_i h_{i-1} t(n-i+1)}{w_{i-1} i(1-t) + w_i h_{i-1} t(n-i+1)}, \\ T_0 = P_0, & T_i = (1-h_i) T_{i-1} + h_i P_i, \end{cases} \tag{2.39}$$

$i = 1, \ldots, n$. From these recursive formulas, we have $P(t) = T_n$. This algorithm also has an elegant geometric interpretation since $T_i \in [T_{i-1}, P_i]$.

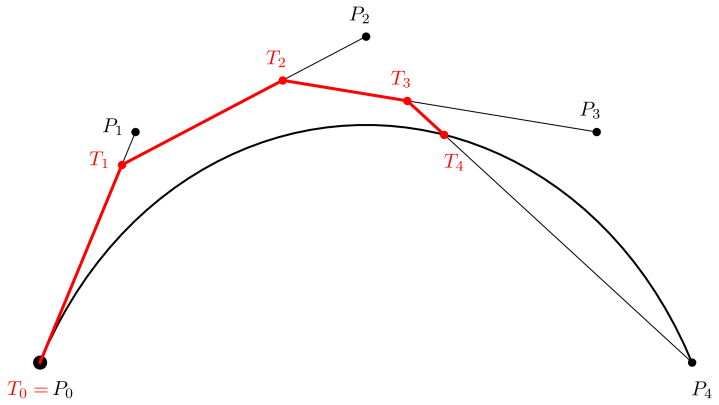


Figure 2.16. Illustration of the algorithm (2.39) for $n = 4$.

2.4.4 Wang–Ball algorithm

Another approach to achieve an algorithm with linear time complexity is by converting the Bernstein basis into a different basis. There exist several methods in this direction, such as transforming the Bernstein into the Wang–Ball basis [37, 83, 120], the DP basis [35, 38, 39], and other similar types of bases [36]; the former is proven to be the most efficient. The rational Wang–Ball curve, defined by the control points $R_0, \ldots, R_n$ with their respective weights $v_0, \ldots, v_n$, is given by

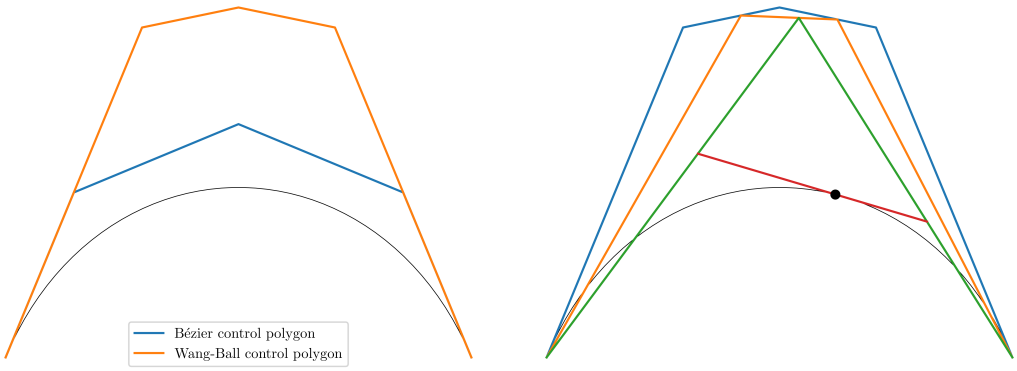
$$P(t) = \frac{\sum_{i=0}^{n} A_i^n(t) v_i R_i}{\sum_{i=0}^{n} A_i^n(t) v_i}, \tag{2.40}$$

where the Wang–Ball basis $\{A_i^n\}_{i=0,\ldots,n}$ is defined as

$$A_i^n(t) = \begin{cases} (2t)^i (1-t)^{i+2}, & 0 \le i \le \lfloor n/2 \rfloor - 1, \\ (2t)^{\lfloor n/2 \rfloor} (1-t)^{\lceil n/2 \rceil}, & i = \lfloor n/2 \rfloor, \\ (2(1-t))^{\lfloor n/2 \rfloor} t^{\lceil n/2 \rceil}, & i = \lceil n/2 \rceil, \\ A_{n-i}^n (1-t), & \lceil n/2 \rceil + 1 \le i \le n. \end{cases} \tag{2.41}$$

Actually, to achieve a linear time method, its implementation uses a recursive algorithm similar to (2.14) (see Figure 2.17), but for the new set of control points and weights (Algorithm 11). Specifically, it starts by setting

$$n_0 = n, \qquad v_i^0 = v_i, \qquad \text{and} \qquad R_i^0 = R_i, \qquad i = 0, \ldots, n^0, \tag{2.42}$$



(a) (b)

Figure 2.17. (a) shows the Wang-Ball control polygon obtained from the Bézier one. (b) illustrates the corner cutting algorithm for evaluating a Wang-Ball curve for $n = 4$.

and then, at each step $r = 1, \dots, n$ of the recursion, it defines $n_r = n - r$ new weights and control points. In particular, if n_r is odd, they are given by

$$\begin{cases} v_i^r = v_i^{r-1}, & i = 0, \dots, \frac{n_r-3}{2}, \\ v_i^r = v_i^{r-1}(1-t) + v_{i+1}^{r-1} t, & i = \frac{n_r-1}{2}, \\ v_i^r = v_i^{r-1}, & i = \frac{n_r+1}{2}, \dots, n_r, \end{cases}$$
$$\begin{cases} R_i^r = R_i^{r-1}, & i = 0, \dots, \frac{n_r-3}{2}, \\ R_i^r = \frac{R_i^{r-1} v_i^{r-1}}{v_i^r}(1-t) + \frac{R_{i+1}^{r-1} v_{i+1}^{r-1}}{v_i^r} t, & i = \frac{n_r-1}{2}, \\ R_i^r = R_i^{r-1}, & i = \frac{n_r+1}{2}, \dots, n_r, \end{cases} \tag{2.43}$$

while, if n_r is even, they are

$$\begin{cases} v_i^r = v_i^{r-1}, & i = 0, \dots, \frac{n_r}{2} - 2, \\ v_i^r = v_i^{r-1}(1-t) + v_{i+1}^{r-1} t, & i = \frac{n_r}{2} - 1, \frac{n_r}{2}, \\ v_i^r = v_i^{r-1}, & i = \frac{n_r}{2} + 1, \dots, n_r, \end{cases}$$
$$\begin{cases} R_i^r = R_i^{r-1}, & i = 0, \dots, \frac{n_r}{2} - 2, \\ R_i^r = \frac{R_i^{r-1} v_i^{r-1}}{v_i^r}(1-t) + \frac{R_{i+1}^{r-1} v_{i+1}^{r-1}}{v_i^r} t, & i = \frac{n_r}{2} - 1, \frac{n_r}{2}, \\ R_i^r = R_i^{r-1}, & i = \frac{n_r}{2} + 1, \dots, n_r, \end{cases} \tag{2.44}$$

and the result is $P(t) = R_0^n$. Before proceeding with this algorithm, there is a pre-processing step to get the values $v_0, \dots, v_n$ and $R_0, \dots, R_n$ (Algorithm 9 and 10).

In particular, the weights and control points of the Bézier and Wang–Ball representations can be converted back and forth by means of a matrix multiplication [68]. However, for the sake of numerical stability, Dejdumrong et al. [37] present the explicit formulas to obtain the Wang–Ball control points and weights from the corresponding Bézier ones, that are

$$\begin{cases} v_0 = w_0, \\ v_n = w_n, \\ v_i = \frac{1}{2^i}\Big[\binom{n}{i}w_i - \sum_{k=0}^{i-1} 2^k\binom{n-2-2k}{i-k}v_k - \sum_{k=n-i+1}^{n} 2^{n-k}\binom{2k-2-n}{k-i}v_k\Big], & i < \lfloor n/2 \rfloor, \\ v_i = \frac{1}{2^{n-i}}\Big[\binom{n}{i}w_i - \sum_{k=0}^{n-i} 2^k\binom{n-2-2k}{i-k}v_k - \sum_{k=i+1}^{n} 2^{n-k}\binom{2k-2-n}{k-i}v_k\Big], & i > \lceil n/2 \rceil, \\ v_i = \frac{1}{2^i}\Big[\binom{n}{i}w_i - \sum_{k=0}^{i-1} 2^k\binom{n-2-2k}{i-k}v_k - \sum_{k=i+2}^{n} 2^{n-k}\binom{2k-2-n}{k-i}v_k\Big], & i = \lfloor n/2 \rfloor, \\ v_i = \frac{1}{2^{n-i}}\Big[\binom{n}{i}w_i - \sum_{k=0}^{i-2} 2^k\binom{n-2-2k}{i-k}v_k - \sum_{k=i+1}^{n} 2^{n-k}\binom{2k-2-n}{k-i}v_k\Big], & i = \lceil n/2 \rceil \end{cases} \tag{2.45}$$

$$\begin{cases} R_0 = P_0, \\ R_n = P_n, \\ R_i = \frac{1}{2^i v_i}\Big[\binom{n}{i}w_i P_i - \sum_{k=0}^{i-1} 2^k\binom{n-2-2k}{i-k}v_k R_k - \sum_{k=n-i+1}^{n} 2^{n-k}\binom{2k-2-n}{k-i}v_k R_k\Big], & i < \lfloor n/2 \rfloor, \\ R_i = \frac{1}{2^{n-i} v_i}\Big[\binom{n}{i}w_i P_i - \sum_{k=0}^{n-i} 2^k\binom{n-2-2k}{i-k}v_k R_k - \sum_{k=i+1}^{n} 2^{n-k}\binom{2k-2-n}{k-i}v_k R_k\Big], & i > \lceil n/2 \rceil, \\ R_i = \frac{1}{2^i v_i}\Big[\binom{n}{i}w_i P_i - \sum_{k=0}^{i-1} 2^k\binom{n-2-2k}{i-k}v_k R_k - \sum_{k=i+2}^{n} 2^{n-k}\binom{2k-2-n}{k-i}v_k R_k\Big], & i = \lfloor n/2 \rfloor, \\ R_i = \frac{1}{2^{n-i} v_i}\Big[\binom{n}{i}w_i P_i - \sum_{k=0}^{i-2} 2^k\binom{n-2-2k}{i-k}v_k R_k - \sum_{k=i+1}^{n} 2^{n-k}\binom{2k-2-n}{k-i}v_k R_k\Big], & i = \lceil n/2 \rceil. \end{cases} \tag{2.46}$$

We note that before computing v_k and R_k, $k = 0, \ldots, n$, the weights v_i and v_{n-i} and the control points R_i and R_{n-i}, $i = 0, \ldots, k-1$, must be computed.

2.4.5 Bernstein–Fourier algorithm

Another series of approaches involving a transformation to another form is explored in [14, 15, 16]; the most efficient amongst them is the Bernstein–Fourier method. It involves applying the Inverse Fast Fourier Transform (IFFT) to the control points, that is, computing the points $\widehat{S}_i = \text{ifft}(\widehat{P}_i)$, $i = 0, \ldots, n$, for $\widehat{P}_i$ in (2.35) (Algorithm 13). Then, $P(t)$ is the central projection on the xy-plane under the projection (2.10) of

$$\widehat{P}(t) = \sum_{i=0}^{n} (\zeta^i t + (1-t))^n \widehat{S}_i, \tag{2.47}$$

where the ζ_i, $i = 0, \ldots, n$, are the roots of unity of order $n+1$. Its implementation (Algorithm 12 and 14) requires $O(n \log n)$ time and involves complex number operations. However, there are some optimisations that can be performed so that this method can compete with the aforementioned methods (Algorithm 12 and 15). First, we note that

$$\widehat{S}_{n+1-i} = \overline{\widehat{S}_i} \qquad \text{for } i = \begin{cases} 1, \ldots, \frac{n}{2}, & \text{if } n \text{ is even,} \\ 1, \ldots, \frac{n-1}{2}, & \text{if } n \text{ is odd.} \end{cases}$$

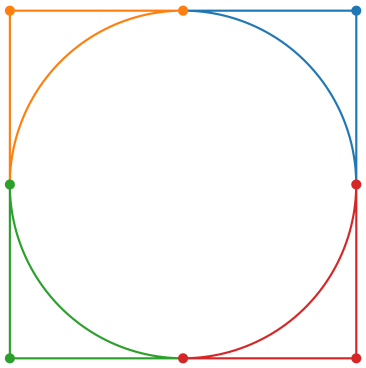


Figure 2.18. A circle is reproduced with four (harmonic) rational quadratic Bézier curves. For each segment, the weights are given by $w_0 = w_2 = 1$ and $w_1 = \sqrt{2}/2$.

Additionally, by letting $s = 1 - t$, we have

$$\overline{(\zeta_i t + (1-t))^n} = \zeta^i (\zeta_i s + (1-s))^n.$$

Hence, we can compute $\widehat{P}(t)$ and $\widehat{P}(1-t)$ simultaneously with $\widehat{S}_k$, $k = 0, \dots, N$, for $N = (n+1)/2$. Then, if n is even, we have

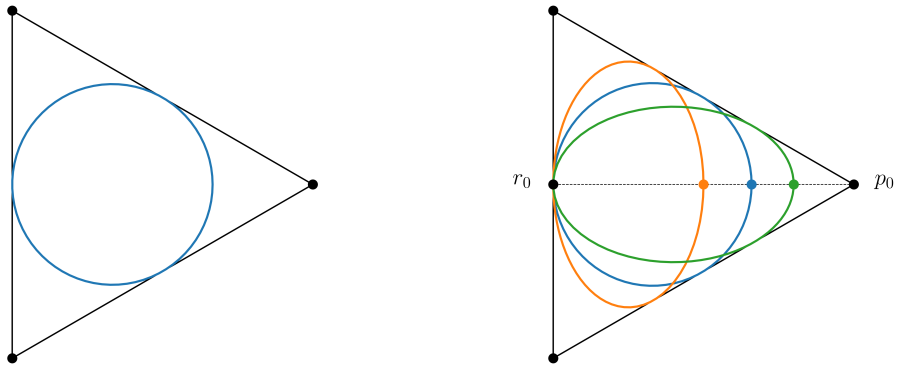
$$\widehat{P}(t) = \widehat{S}_0 + 2\sum_{i=1}^{n/2} \mathrm{Re}\big((\zeta_i t + (1-t))^n \widehat{S}_i\big),$$
$$\widehat{P}(1-t) = \widehat{S}_0 + 2\sum_{i=1}^{n/2} \mathrm{Re}\left((\zeta_i t + (1-t))^n \frac{\overline{\widehat{S}_i}}{\overline{\zeta_i}}\right),$$

while, if n is odd, we have

$$\widehat{P}(t) = \widehat{S}_0 - (1-2t)^n \widehat{S}_N + 2\sum_{i=1}^{N-1} \mathrm{Re}\big((\zeta_i t + (1-t))^n \widehat{S}_i\big),$$
$$\widehat{P}(1-t) = \widehat{S}_0 - (1-2t)^n \widehat{S}_N + 2\sum_{i=1}^{N-1} \mathrm{Re}\left((\zeta_i t + (1-t))^n \frac{\overline{\widehat{S}_i}}{\overline{\zeta_i}}\right).$$

2.5 Closed curves

Although Bézier curves are the standard models for curve creation, creating closed curves with a single Bézier curve, especially while maintaining high-order



(a) (b)

Figure 2.19. A circle can be reproduced with periodic Bézier curve given by regular polygons (a). The effect of changing the weight w_0 by dragging the point $p(0)$ on the segment $[p_0, r_0]$.

continuity at the end-points, can be challenging. We need at least 6 control points to reproduce a circle with a single rational Bézier curve [31, 103]. The most popular methods are by using interpolating splines [28, 126, 127, 129] or NURBS [44, 87]. The conic sections can be reproduced with splines so that each section is a quadratic rational curve (see Figure 2.18).

Free-form closed curves are harder to model with rational spline curves while trying to achieve high-order continuity at the joints. To remedy that, we can use trigonometric polynomial curves to design rational spline (closed) curves, as they are C^∞ smooth curves and can be reproduced with rational splines [104]. Each section is given by a special class of rational curves called *harmonic rational Bézier curves* [56].

2.5.1 Periodic basis

We recall that a trigonometric polynomial of order N is of the form

$$p(t) = p_0 + \sum_{k=1}^{N} [p_{2k-1} \cos kt + p_{2k} \sin kt], \tag{2.48}$$

But as for Bézier curves (2.4), ideally, we want a closed form that produce affine invariant curves, that is,

$$p(t) = \sum_{i=0}^{n} B_i^n(t) p_i, \qquad t \in [0, 2\pi] \tag{2.49}$$

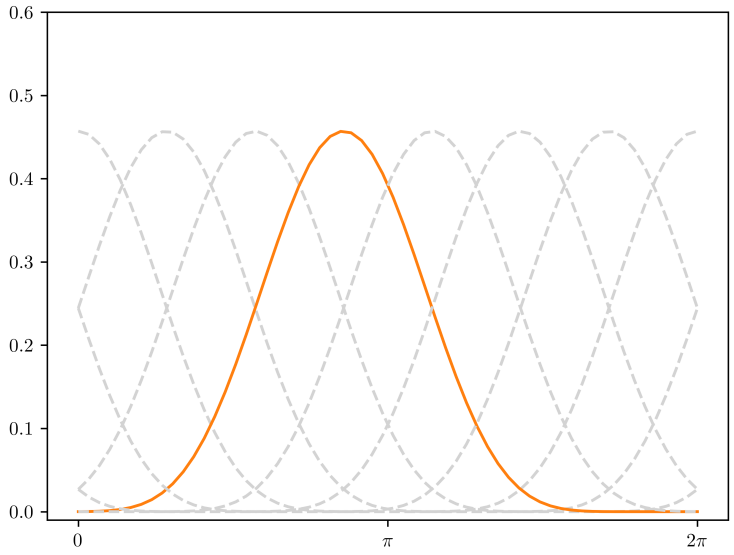


Figure 2.20. The shape of the periodic basis functions for $n = 6$.

where the functions B_i^n are 2π-periodic trigonometric functions and form a basis of space trigonometric polynomials $\{1, \cos t, \sin t, \ldots, \cos Nt, \sin Nt\}$. Note that we keep the same notation as for the basis functions for the open case in (2.4). The definition simply changes according to the context whether we are in the open or the closed setting. These spaces are spanned by an odd number of basis functions, so we have to restrict ourselves to the case where $n = 2N$. Furthermore, for closed curves, there is no notion of endpoint, which means that points can be relabelled and any point p_i can be considered as p_0. This motivates the fact that the functions $B_i^n(t)$ are uniform shifts of one another, that is,

$$B_i^n(t) = B_n(t - \phi_i), \qquad \phi_i = \frac{2\pi i}{n+1}, \tag{2.50}$$

for some function $B_n(t)$.

Sánchez-Reyes [105] proposes a suitable requirements for the kernel B to mimic the central N-th Bernstein polynomial (see Figure 2.20):

$\mathbf{p}_1$ - non-negativity: $B_n(t) \geq 0$,

$\mathbf{p}_2$ - symmetry: $B_n(t) = B_n(-t)$,

$\mathbf{p}_3$ - partition of unity: $\sum_{i=0}^{n} B_n(t - \phi_i) = 1$ for $t \in [0, 2\pi]$,

$\mathbf{p}_4$ - bell-shape: it is monotonic increasing in $(-\pi, 0)$, decreasing on $(0, \pi)$, and attaining its maximal at $t = 0$. It also has a contact of order n with the

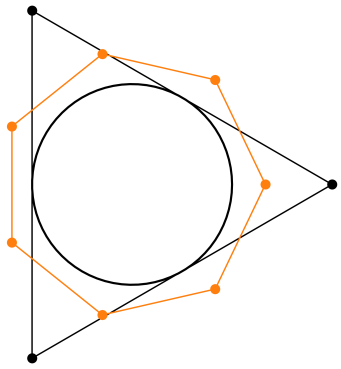


Figure 2.21. Rasing the number of control points from 3 to 7.

x–axis at $\pm\pi$. That is

$$\left.\frac{d^j B_n(t)}{dt^j}\right|_{\pm\pi} = 0, \quad \text{for } j = 0, \ldots, n.$$

Róth et al. [97], Sánchez-Reyes [105] show that

$$B_n(t) = \frac{2^n}{(n+1)}\binom{n}{N}^{-1} \cos^n \frac{t}{2} \tag{2.51}$$

is a suitable candidate and that the functions $B_i(t)$ form a basis of the space of trigonometric polynomials. The curves determined by this basis are called *periodic Bézier curves*. Akin to Bézier curves, the construction extends naturally to the rational setting [70], and they are also the image under the projection Π in (2.10) of a spatial periodic Bézier curves. The manipulations described in 2.3.1, 2.3.2, 2.3.3, 2.3.4 translate naturally to the closed curve setting (see Figure 2.19.b).

2.5.2 Degree elevation

Degree elevation of periodic Bézier curves from $n+1$ to $\widetilde{n}+1$ number of points (see Figure 2.21), $\widetilde{n} = 2\widetilde{N}$, can be done in one step involving Fast Fourier Transforms (FFT) [119]. However, it boils down to a single matrix multiplication. Let $\boldsymbol{p}_n = \{p_k\}_{k=0,\ldots,n}$, $\widetilde{\boldsymbol{p}}_{\widetilde{n}} = \{\widetilde{p}_k\}_{k=0,\ldots,\widetilde{n}}$, then $\widetilde{\boldsymbol{p}}_{\widetilde{n}} = A\boldsymbol{p}_n$ where

$$A = \left(a_{ij}\right)_{0\le i\le\widetilde{n},\ 0\le j\le n}, \qquad a_{ij} = \frac{1}{n+1}\sum_{k=-N}^{N} \frac{\zeta_{\widetilde{n}}^{ik}}{\zeta_n^{jk}} \frac{f_{k,N}}{f_{k,\widetilde{N}}},$$

$\zeta_n = e^{\frac{2\pi i}{n+1}}$, $\zeta_{\widetilde{n}} = e^{\frac{2\pi i}{\widetilde{n}+1}}$, and $f_{j,N} = \prod_{i=1}^{|j|} \frac{N-i+1}{N-i}$, and $f_{j,\widetilde{N}} = \prod_{i=1}^{|j|} \frac{\widetilde{N}-i+1}{\widetilde{N}-i}$. For rational curves, the degree elevation involves extra-steps of conversion to spatial polynomial curve and applying degree elevation on this latter.

Chapter 3

Barycentric rational curves

Interpolation is a method for estimating intermediate data values given a set of known data points. Since our work focusses on curve design, especially, curves that are defined by a parametric rational Bézier representation, we are interested in C^∞ smooth interpolating functions.

3.1 Neville's algorithm

The de Casteljau algorithm (2.3) is one natural extension of the linear interpolation (2.2). But the resulting curve does not interpolate the points except for p_0 and p_n. Neville [80] describes another natural extension that results in a curve that interpolates each point p_k at a certain respective parameter t_k. Assume that we are given a set of distinct parameters $t_0, \ldots, t_n$. A generalisation of the linear interpolation (2.2) such that $p(t_i) = p_i$ and $p(t_{i+1}) = p_{i+1}$ is given by

$$p_i^1(t) = p_i \frac{t - t_{i+1}}{t_i - t_{i+1}} + \frac{t - t_i}{t_{i+1} - t_i} p_{i+1}. \tag{3.1}$$

The *Neville's algorithm* [66] generalises (3.1) to obtain a curve that satisfies $p(t_k) = p_k$,

$$\begin{array}{ll} p_i^0 = p_i, & i = 0, \ldots, n \\ p_i^j = p_i^{j-1} \frac{t - t_{i+j}}{t_i - t_{i+j}} + \frac{t - t_i}{t_{i+j} - t_i} p_{i+1}^{j-1}, & i = 0, \ldots, n-j,\ j = 1, \ldots, n. \end{array} \tag{3.2}$$

In this case, the curve is also given by $p(t) = p_0^n$.

3.2 Lagrange interpolation

As for Bézier curves (2.4), we can write $p(t)$ as a linear combination

$$p(t) = \sum_{i=0}^{n} \ell_i(t) p_i, \tag{3.3}$$

such that $\ell_k(t)$ is a polynomial of degree n. ℓ_k is the output of the recursive algorithm (3.2) by taking $p_k^0 = 1$ and $p_i^0 = 0$ for $i \neq k$. It is clear that $\ell_k(t)$ satisfies the *Lagrange property*

$$\ell_k(t_i) = \begin{cases} 1, & \text{if } k = i, \\ 0, & \text{otherwise,} \end{cases} \qquad k, i = 0, \ldots, n. \tag{3.4}$$

From the Lagrange property, we can devise a simple expression of $\ell_k(t)$. Since $\ell_k(t)$ is a polynomial of degree n and that $\ell_k(t_i) = 0$ for $k \neq i$. This means that all t_i, $i \neq k$ are n distinct roots of $\ell_k(t)$. As a consequence of the fundamental theorem of algebra [4],

$$\ell_k(t) = \omega_k \prod_{i=0,\, i\neq k}^{n} (t - t_i) \tag{3.5}$$

for some non-zero scalar ω_k. The fact that $\ell_k(t_k) = 1$ implies that

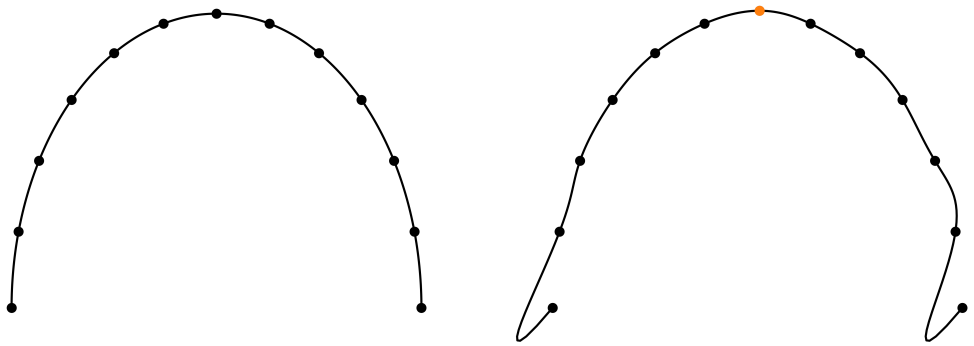
$$\omega_k = \prod_{j=0,\, j\neq k}^{n} \frac{1}{t_k - t_j}. \tag{3.6}$$

A such scalar ω_k is called *a Lagrange weight*, $\ell_k(t)$ in (3.5) is called *a Lagrange function*, and (3.3) is called *Lagrange interpolation*.

3.3 Vandermonde matrix

We can also recover the Lagrange functions from a Vandermonde matrix. The function $\ell_k(t)$ is a polynomial of degree n. We can express it in monomial form as $\ell_k(t) = a_0 + a_1 t + \cdots + a_n t^n$ for some $a_0, \ldots, a_n \in \mathbb{R}$. The Lagrange properties (3.4) can be written compactly as $V(t_0, \ldots, t_n)\boldsymbol{a} = \boldsymbol{e}_k$ where $\boldsymbol{a} = (a_0, \ldots, a_n)^T$, $\boldsymbol{e}_k = (0, \ldots, 1, \ldots, 0)^T$, and $V(t_0, \ldots, t_n)$ is the Vandermonde matrix

$$V(t_0, \ldots, t_n) = \begin{pmatrix} 1 & t_0 & \cdots & t_0^n \\ \vdots & \ddots & & \vdots \\ 1 & t_n & \cdots & t_n^n \end{pmatrix}.$$



(a) (b)

Figure 3.1. (a) shows a Lagrange polynomial curve of a half-ellipse sampled at 13 points, with respect to the equidistant nodes $t_i = \frac{i}{12}$. (b) shows the effect of perturbing q_6 by $(0.01, 0.01)$.

If the parameters $t_0, \ldots, t_n$ are pairwise distinct, then we have $\boldsymbol{a} = V(t_0, \ldots, t_n)^{-1} e_k$. By multiplying each side with the row vector $(1, t, \ldots, t^n)$ on each side, we conclude that

$$\ell_k(t) = \frac{\det V(\ldots, t_{k-1}, t, t_{k+1}, \ldots)}{\det V(t_0, \ldots, t_n)}. \tag{3.7}$$

Since the input parameters are pairwise distinct, the determinant of a Vandemonde matrix [29, 128] is given by

$$\det V(t_0, \ldots, t_n) = \prod_{0 \le i < j \le n} (t_i - t_j). \tag{3.8}$$

By combining (3.7) and (3.8) and after cancelling some common factor, we get the Lagrange functions as in (3.5).

3.4 Limitation of Lagrange interpolation

High-degree Lagrange polynomials tend to oscillate excessively near endpoints, especially for uniformly distributed nodes. This is known as *Runge's phenomenon* [98] (see Figure 3.2). This poses a problem for the use of Lagrange polynomial in curve design since even small manipulations can lead to huge unwanted artefacts near the end-points (see Figure 3.1).

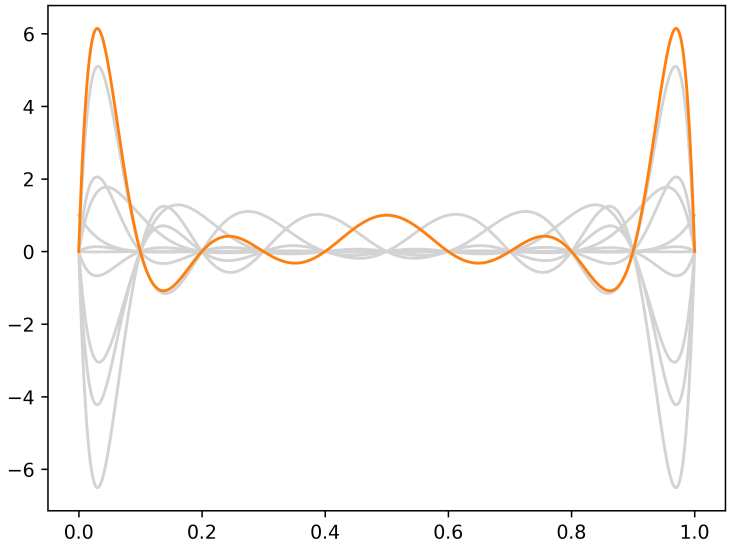


Figure 3.2. This illustrates the Lagrange basis functions $\ell_i(t)$, $i = 0, \ldots, 12$, with emphasis on ℓ_6, with respect to an equidistant set of nodes $t_i = \frac{i}{12}$. We observe the strange behaviour of $\ell_6(t)$ which leads to huge artifact in Figure 3.1 near the endpoints.

3.5 Barycentric rational interpolation

The Lagrange interpolant (3.3) can be written in *first barycentric form* [99] as

$$p(x) = \ell(t) \sum_{i=0}^{n} \frac{\omega_i}{t - t_i} p_i, \tag{3.9}$$

where $\ell(t) = (t - t_0) \cdots (t - t_n)$. The alternative, a rational form, is obtained by dividing (3.9) by the interpolant of the constant function 1 written in the first barycentric form as $1 = \ell(t) \sum_{i=0}^{n} \frac{\omega_i}{t - t_i}$. We write the *second barycentric form* as

$$p(t) = \frac{\sum_{i=0}^{n} \frac{\omega_i}{t - t_i} p_i}{\sum_{i=0}^{n} \frac{\omega_i}{t - t_i}}. \tag{3.10}$$

This form of a weighted sum can be traced back to [40, 118] where it was referred to as *normalized Lagrangian interpolation*, praised for its fast evaluation.

Assume we replace the weights ω_i by random scalars v_i, that is,

$$p(t) = \frac{\sum_{i=0}^{n} \frac{v_i}{t - t_i} p_i}{\sum_{i=0}^{n} \frac{v_i}{t - t_i}}. \tag{3.11}$$

Multiplying both the numerator and the denominator by $\ell(t)$, we see that we obtain a generic rational curve. Conversely, [12] shows that every rational function can be written in barycentric form. In fact, consider a rational function

$$p(t) = \frac{N(t)}{D(t)}, \tag{3.12}$$

for some degree n polynomials $N(t)$ and $D(t)$. We first express D and $N = D \cdot p$ in the first barycentric form (3.9) with respect to the same nodes.

$$N(t) = \ell(t) \sum_{i=0}^{n} \frac{\omega_i}{t - t_i} D(t_i) p(t_i), \tag{3.13}$$

$$D(t) = \ell(t) \sum_{i=0}^{n} \frac{\omega_i}{t - t_i} D(t_i). \tag{3.14}$$

Then we recover the barycentric form (3.11) by writing the fraction $\frac{N}{D}$ and cancelling out some factors. Furthermore, we have

$$p_i = p(t_i), \qquad v_i = \omega_i D(t_i), \qquad i = 0, \ldots, n. \tag{3.15}$$

For some particular set of nodes, the Lagrange weights are given by explicit formulas [13], making the conversion simpler. For uniform nodes, $t_k = t_0 + kh$ for some non-zero scalar h, the weights are given by $\omega_k = \frac{(-1)^{n+k}}{h^n n!}\binom{n}{k}$ [111], which after cancelling the common factor $(-1)^n h^n n!$ becomes

$$\omega_k = (-1)^k \binom{n}{k}. \tag{3.16}$$

We note that the Lagrange weights ω_k can also be defined as [62]

$$\omega_k = \frac{1}{\ell'(x_k)}. \tag{3.17}$$

This induces a simple formula for the weights with respect to the Chebyshev nodes of the first kind $t_k = \cos\frac{(2k+1)\pi}{2n+2}$ and the Chebyshev nodes of the second kind $t_k = \cos\frac{k\pi}{n}$, respectively as [102]

$$\omega_k = (-1)^k \sin\frac{(2k+1)\pi}{2n+2}, \quad \omega_k = (-1)^k \delta_k, \quad \delta_k = \begin{cases} 1/2 & k = 0 \text{ or } k = n \\ 1 & \text{otherwise.} \end{cases} \tag{3.18}$$

3.5.1 Interpolation without poles

It might occur that a curve given by (3.11) is discontinuous. This is not ideal, especially in the context of curve design. Hence, it would be ideal to find some constraints on the weights v_k so that $p(t)$ has no poles. From an observation in [108], the points p_i are interpolated if and only if the weights v_k are nonzero for all k. From (3.16) and (3.18), we observe that the weights have alternating signs. In fact, for design purposes, this is necessary [108].

Proposition 3.1. *Consider a rational curve* (3.11). *If the curve is continuous in* $[t_0, t_n]$*, then the signs of the weights* v_i *alternate.*

Proof. By writing $p(t) = N(t)/D(t)$, the denominator $D(t)$ must not have a zero in $[t_0, t_n]$ for the curve p to be continuous. This means that $D(t_i)$ have the same signs for $i = 0, \dots, n$. Furthermore, from (3.6), we have $\operatorname{sgn} \omega_i = (-1)^{n-i}$. Henceforth, by (3.15), we have $\operatorname{sgn} v_i = (-1)^{n-i} \operatorname{sgn} D(t_i)$, which proves the proposition. □

Berrut [11] states that the interpolation problem is well-conditioned if the weights have the same absolute values. By combining that with Proposition!3.1, [11] deduced that a barycentric interpolant with a particular set of weight, called *Berrut weights*, where $v_k = (-1)^k$ for all k is well-conditioned and has no poles. We also know that by using the Lagrange weights ω_k in (3.6), we have a polynomial interpolant. These weights are generalised by [49] to obtain the *Floater–Hormann* weights

$$\omega_i = (-1)^i \sum_{j=\max(i-d,0)}^{\min(i,n-d)} \prod_{k=j,k\neq i}^{j+d} \frac{1}{|t_i - t_k|}. \tag{3.19}$$

For these very specific weights, barycentric rational interpolations are guaranteed to have no poles and a high approximation order [11, 49], with slow-growing Lebesgue constants, in particular for equidistant nodes [22]. Apart from curve design, which is the main focus of this thesis, the barycentric form is also a key ingredient of the AAA algorithm [79], which extends the work of Antoulas and Anderson [3] and uses an adaptive node selection scheme to efficiently compute robust rational approximations of real and complex functions.

3.5.2 Derivatives of barycentric rational forms

An advantage of the barycentric form is that the forms of its derivatives are very simple formulas [108].

Proposition 3.2. *Consider a barycentric rational interpolant* (3.11). *We have*

- *if* $t \neq t_j$, $j = 0, \ldots, n$ *and* $k \geq 0$, *we have*

$$\frac{p^{(k)}(t)}{k!} = \frac{\sum_{i=0}^{n} \frac{v_i}{t-t_i} p[(t)^k, t_i]}{\sum_{i=0}^{n} \frac{v_i}{t-t_i}}, \tag{3.20}$$

- *and if* $t = t_j$, *and* $k \geq 1$

$$\frac{p^{(k)}(t_j)}{k!} = -\frac{\sum_{i=0, i\neq j}^{n} v_i p[(t_j)^k, t_i]}{v_j}. \tag{3.21}$$

where $p[(t)^k, t_i] = p[t, \ldots, t, t_i]$ *is a divided difference.*

Proof. We prove (3.20) by induction in k. For $k = 0$, (3.20) reduces to (3.11). Now, assume that (3.20) is true for $k \geq 0$. We recall that [34]

$$\frac{p^{(k)}(t)}{k!} = p[(t)^{k+1}]. \tag{3.22}$$

Hence from (3.20) and (3.22) we have

$$\sum_{i=0}^{n} v_i \frac{p[(t)^k, t_i] - p[(t)^{k+1}]}{t - t_i} = 0.$$

This means that

$$\sum_{i=0}^{n} v_i p[(t)^{k+1}, t_i] = 0. \tag{3.23}$$

Furthermore, a differentiation with respect to t on both sides of (3.23) yields

$$(k+1) \sum_{i=0}^{n} v_i p[(t)^{k+2}, t_i] = 0$$

On the left hand side we have

$$\sum_{i=0}^{n} v_i p[(t)^{k+2}, t_i] = \sum_{i=0}^{n} v_i \frac{p[(t)^{k+1}, t_i] - p[(t)^{k+2}]}{t - t_i}.$$

It is clear from here that (3.20) holds for $k + 1$.

Now, by substituting $t = t_j$ in (3.23) yields

$$v_j p[(t)^{k+2}] + \sum_{i=0, i\neq j}^{n} v_i p[(t_j)^{k+1}, t_i] = 0.$$

It is clear from here that (3.20) holds. □

3.6 Trigonometric barycentric rational interpolation

The barycentric form (3.11) has its analogue for trigonometric rational functions. In the trigonometric setting, we assume $0 \le t_0 < \cdots < t_n < 2\pi$. As for the periodic Bézier curves, assume $n = 2N$ for some N. First, let us see how to write a polynomial curve in barycentric form. It is well known that there exists a polynomial of the form [64, 113]

$$p(t) = \sum_{k=-N}^{N} c_k e^{\mathrm{i}kt}, \qquad c_k = \frac{1}{n+1} \sum_{j=0}^{n} p_j e^{-\mathrm{i}jt_k},$$

that satisfies $p(t_i) = p_i$. We can write $p(t)$ as in (3.3) where

$$\ell_j(t) = \frac{1}{n+1} \sum_{k=-N}^{N} e^{\mathrm{i}kt} e^{-\mathrm{i}jt_k}.$$

It is a geometric series with ratio $e^{\mathrm{i}(t-t_j)}$ and initial term $e^{-\mathrm{i}N(t-t_j)}$. By considering uniformly distributed nodes $t_k = 2k\pi/(n+1)$, we have [63]

$$\begin{aligned}
\ell_j(t) &= \frac{1}{n+1} \frac{e^{\mathrm{i}(n+1)t} - 1}{e^{\mathrm{i}(t-t_j)} - 1} e^{-\mathrm{i}N(t-t_j)} \\
&= \frac{1}{n+1} \frac{e^{\mathrm{i}(n+1)t} - 1}{e^{\mathrm{i}Nt}} \frac{e^{\mathrm{i}(N+1)t_j}}{e^{it} - e^{it_j}} \\
&= \frac{(-1)^j}{n+1} \sin\frac{(n+1)t}{2} \csc\frac{t-t_j}{2}.
\end{aligned}$$

Therefore, $p(t)$ can be written as

$$p(t) = \frac{1}{n+1} \sin\frac{(n+1)t}{2} \sum_{j=0}^{n} (-1)^j \csc\frac{t-t_j}{2} p_j.$$

By dividing with the constant polynomial $1 = \frac{1}{n+1} \sin\frac{(n+1)t}{2} \sum_{j=0}^{n} (-1)^j \csc\frac{t-t_j}{2}$, we write $p(t)$ in barycentric form as

$$p(t) = \frac{\sum_{j=0}^{n} (-1)^j \csc\frac{t-t_j}{2} p_j}{\sum_{j=0}^{n} (-1)^j \csc\frac{t-t_j}{2}}.$$

For more general set of nodes, the *Gauss's formula for trigonometric interpolation* states that

$$p(t) = \sum_{i=0}^{n} \ell_i(t) p_i, \qquad \ell_i(t) = \prod_{j=0,\, j\neq i}^{n} \frac{\sin\frac{t-t_j}{2}}{\sin\frac{t_i-t_j}{2}}. \tag{3.24}$$

By factoring $\ell(t) = \prod_{j=0}^{n} \sin \frac{t-t_j}{2}$, we have the trigonometric analogue of the first barycentric form (3.9) as [10, 100]

$$p(t) = \ell(t) \sum_{i=0}^{n} \omega_i \csc \frac{t-t_i}{2} p_i \tag{3.25}$$

where

$$\omega_i = \prod_{j=0,\, j\neq i}^{n} \csc \frac{t_i - t_j}{2}, \qquad i = 0, \ldots, n. \tag{3.26}$$

Dividing by the constant function $1 = \ell(t) \sum_{i=0}^{n} \omega_i \csc \frac{t-t_i}{2}$. A polynomial can be written in barycentric form as

$$p(t) = \frac{\sum_{i=0}^{n} \omega_i \csc \frac{t-t_i}{2} p_i}{\sum_{i=0}^{n} \omega_i \csc \frac{t-t_i}{2}}.$$

Here again, by taking a random weight v_i instead of ω_i, we recover a generic rational curve. The converse is also true. In fact, consider a rational curve $p(t) = N(t)/D(t)$. By writing them in barycentric rational form

$$N(t) = \frac{\sum_{i=0}^{n} \omega_i \csc \frac{t-t_i}{2} N(t_i)}{\sum_{i=0}^{n} \omega_i \csc \frac{t-t_i}{2}}, \qquad D(t) = \frac{\sum_{i=0}^{n} \omega_i \csc \frac{t-t_i}{2} D(t_i)}{\sum_{i=0}^{n} \omega_i \csc \frac{t-t_i}{2}},$$

and by noticing that $N(t) = D(t)p(t)$, we can write $p(t)$ in barycentric form

$$p(t) = \frac{\sum_{i=0}^{n} (-1)^i \csc \frac{t-t_i}{2} v_i p_i}{\sum_{i=0}^{n} (-1)^i \csc \frac{t-t_i}{2} v_i}, \tag{3.27}$$

where

$$p_i = p(t_i), \qquad v_i = (-1)^i \omega_i D(t_i), \qquad i = 0, \ldots, n. \tag{3.28}$$

Akin for the classical case, we can use the same reasoning as in Proposition 3.1 to observe that to have a continuous curve, i.e. $p(t)$ has no pole, it is necessary that the weights v_i have the same signs.

Proposition 3.3. *Consider a trigonometric barycentric rational curve* (3.27). *If $p(t)$ is free of poles, then the weights $v_0, \ldots, v_n$ have the same sign.*

Proof. The assertion follows easily from (3.28) and the fact that $\operatorname{sgn} \omega_i = (-1)^i$. □

Chapter 4

Shape control techniques for rational Bézier curves

In Section 2.3.10 we note that there exist different types of control polygon that can be used to manipulate rational Bézier curves. One of them being a control polygon such that the vertices are interpolated. In this chapter, we discuss the shape control tools induced by the barycentric rational form (3.11). This chapter is a recollection of our work in [93].

To distinguish between Bézier points and interpolation points, we denote rational Bézier pairs as (p_i, w_i) and interpolation data pairs as (q_i, v_i) for $i = 0, \ldots, n$. Moreover, in order to make emphasis on the fact that the signs of the barycentric weights are alternating, we assume $v_i > 0$, $i = 0, \ldots, n$ and write the curve as

$$q(t) = \frac{\sum_{i=0}^{n} \frac{(-1)^i}{t-t_i} v_i q_i}{\sum_{i=0}^{n} \frac{(-1)^i}{t-t_i} v_i}. \tag{4.1}$$

Furthermore, we write $p(t)$ and $q(t)$ to distinguish the type of parametrisation, whether it is in a rational Bézier form (2.8) or in a barycentric rational form (4.1), respectively.

4.1 Equivalence of Bézier and barycentric form

Let us first recall that a rational Bézier form is a projection of spatial curve under the projection (2.10), hence $p(t)$ can be written $p(t) = (\widehat{x}(t), \widehat{y}(z))/\widehat{z}(t)$ as in (3.12). Hence by (3.15), a rational Bézier curve can be parametrised in barycentric form (4.1) such that

$$q_i = p(t_i), \qquad v_i = (-1)^{n+i} \omega_i z_i, \qquad z_i = \widehat{z}(t_i), \tag{4.2}$$

for $i = 0, \dots, n$, and ω_i is a Lagrange weight as in (3.6). Note that $\operatorname{sgn} \omega_i = (-1)^{n-i}$, which means that $v_i > 0$.

In principle, the nodes t_i do not have to be ordered or restricted to the interval $[0, 1]$, as long as they are distinct; however, in the context of interactive curve design, it seems natural to make these assumptions. Likewise, it is reasonable to set $t_0 = 0$ and $t_n = 1$, so that $q_0 = p_0$ and $q_n = p_n$ mark the endpoints of the curve.

A natural question to ask at this point is: how does one get back from barycentric to Bézier form? To this end, it helps to recall that $\widehat{p}_i = (w_i p_i, w_i)$ and to let $\widehat{q}_i = (z_i q_i, z_i)$, so that the assignments $q_i = p(t_i)$ and $z_i = \widehat{z}(t_i)$ for $i = 0, \dots, n$ can be written compactly as $\widehat{\boldsymbol{Q}} = \boldsymbol{B}\widehat{\boldsymbol{P}}$, where[1]

$$\boldsymbol{B} = \begin{pmatrix} B_0^n(t_0) & \cdots & B_n^n(t_0) \\ \vdots & \ddots & \vdots \\ B_0^n(t_n) & \cdots & B_n^n(t_n) \end{pmatrix}, \qquad \widehat{\boldsymbol{P}} = \begin{pmatrix} \widehat{p}_0 \\ \vdots \\ \widehat{p}_n \end{pmatrix}, \qquad \widehat{\boldsymbol{Q}} = \begin{pmatrix} \widehat{q}_0 \\ \vdots \\ \widehat{q}_n \end{pmatrix}. \tag{4.3}$$

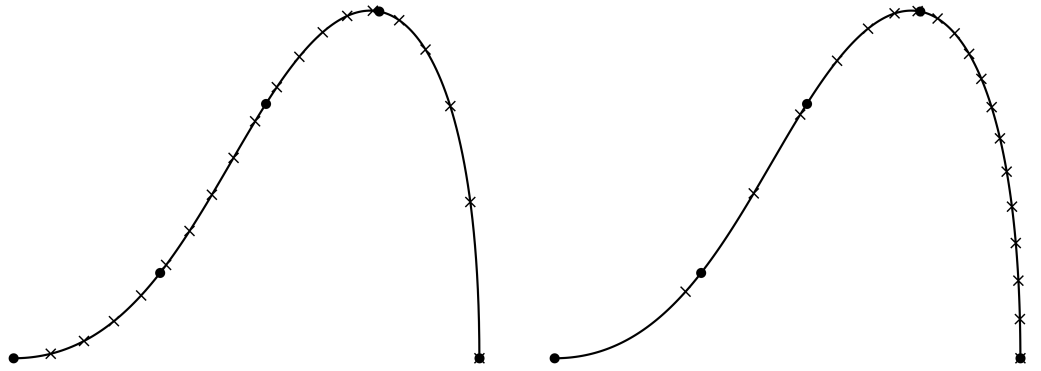
Here, $B_i^n(t)$ is a Bernstein polynomial of degree n as in (2.6).

Proposition 4.1. *The barycentric rational curve* (4.1) *with nodes t_i, interpolation points q_i, and weights v_i can be expressed in Bézier form* (2.8) *with control points $p_i = (\widehat{x}_i, \widehat{y}_i)/\widehat{z}_i$ and weights $w_i = \widehat{z}_i$, where the vector $\widehat{\boldsymbol{P}}$ of points $\widehat{p}_i = (w_i p_i, w_i) = (\widehat{x}_i, \widehat{y}_i, \widehat{z}_i)$ is defined as $\widehat{\boldsymbol{P}} = \boldsymbol{B}^{-1}\widehat{\boldsymbol{Q}}$ and $\widehat{\boldsymbol{Q}}$ is the vector of points $\widehat{q}_i = (z_i q_i, z_i)$ with $z_i = (-1)^{n+i} v_i/\omega_i$ and ω_i as in* (3.6).

Proof. The assertion follows easily from the fact that the *Bernstein–Vandermonde* matrix $\boldsymbol{B}$ in (4.3) is non-singular, because the Bernstein basis is a Chebyshev system. □

Note that $\widehat{\boldsymbol{P}} = \boldsymbol{B}^{-1}\widehat{\boldsymbol{Q}}$ can be computed fast and accurately with $O(n^2)$ time complexity [75]. If the last coordinate $w_i = \widehat{z}_i$ of the homogeneous control point $\widehat{p}_i$ happens to vanish for some i, it means that the given barycentric rational curve cannot be written as a classical rational Bézier curve with control points in $\mathbb{R}^2$ with the same degree. Instead, the control point p_i needs to be replaced by the *control vector* $w_i p_i = (\widehat{x}_i, \widehat{y}_i) \in \mathbb{R}^2$, representing an *infinite* control point in this case [45, 85]. Alternatively, since we assume that the curve is continuous, $\widehat{z}(t) > 0$, $t \in [0, 1]$. Moreover, degree elevation algorithms (2.32) produce piecewise linear approximations of Bézier curves that progressively converge to the original curve [78, 91]. Hence we can degree elevate the spatial curve one or few times, and eventually, all rational Bézier weights will be non-zero.

[1] The observant reader may have already noticed that throughout this paper we write *points* (in $\mathbb{R}^2$ and $\mathbb{R}^3$) as *row vectors*, so as to avoid excessive use of the transposition operator and to be able to conveniently stack them into matrices, like $\widehat{\boldsymbol{P}}, \widehat{\boldsymbol{Q}} \in \mathbb{R}^{(n+1)\times 3}$.



(a) (b)

Figure 4.1. This illustrates the difference between two parametrisations of the same curve (3.11) with parameters $t_k = k/4$ and weights $v_k = (n+1-k)\omega_k$ where ω_k is a Lagrange weight (3.6) (a) and $t_0 = 1$, $t_1 = 0.0625$, $t_2 \approx 1.66$, $t_3 = 0.375$, $t_4 = 1$, with weights $v_0 = v_5 = 1$, $v_1 = v_3 = -4$, $v_2 = 6$ (b). The crosses visualise the curve points $p(k/20)$, $k = 1, \ldots, 19$.

4.2 Standard form

Once a rational curve is represented in barycentric form, we show in [93] that we can use a linear rational reparametrisation to bring it into *standard barycentric form* with $v_0 = v_n = 1$, very similar to how the Bézier representation can be brought into standard form [82].

Lemma 4.2. *For any $\lambda \in (0,1)$, consider the linear rational reparameterization $\varphi : [0,1] \to [0,1]$,*

$$\varphi(t) = \frac{(1-\lambda)t}{\lambda(1-t)+(1-\lambda)t}. \tag{4.4}$$

Let p be the barycentric rational curve (3.10) *with nodes t_i, interpolation points p_i, and weights v_i and let $\widetilde{p}$ be the barycentric rational curve with nodes $\widetilde{t}_i = \varphi(t_i)$, the same interpolation points $\widetilde{p}_i = p_i$, and weights $\widetilde{v}_i = v_i \lambda \widetilde{t}_i / t_i$. Then, $p = \widetilde{p} \circ \varphi$.*

Proof. Denoting the denominator of $\varphi(t)$ by $\delta(t) = \lambda(1-t)+(1-\lambda)t$, we first observe that

$$\frac{\widetilde{t}_i}{\varphi(t)-\widetilde{t}_i} = \frac{\varphi(t_i)}{\varphi(t)-\varphi(t_i)} = \frac{\frac{(1-\lambda)t_i}{\delta(t_i)}}{\frac{(1-\lambda)t}{\delta(t)} - \frac{(1-\lambda)t_i}{\delta(t_i)}} = \frac{t_i\delta(t)}{t\delta(t_i)-t_i\delta(t)} = \frac{t_i\delta(t)}{\lambda(t-t_i)}$$

for any $i = 0, \dots, n$. Therefore,

$$\frac{\widetilde{v}_i}{\varphi(t) - \widetilde{t}_i} = \frac{v_i \lambda}{t_i} \cdot \frac{\widetilde{t}_i}{\varphi(t) - \widetilde{t}_i} = \frac{v_i}{t - t_i} \delta(t).$$

After substituting this into the numerator and the denominator of $(\widetilde{p} \circ \varphi)(t)$ and cancelling the common factor $\delta(t)$ we get $(\widetilde{p} \circ \varphi)(t) = p(t)$. □

Note that $\widetilde{v}_0$ in Lemma 4.2 is well defined, even if $t_0 = 0$, because

$$\lim_{t \to 0} \frac{\varphi(t)}{t} = \lim_{t \to 0} \varphi'(t) = \frac{1 - \lambda}{\lambda},$$

so that $\widetilde{v}_0 = v_0(1 - \lambda)$ in that case. Moreover, we observe that $\operatorname{sign}(\widetilde{v}_i) = \operatorname{sign}(v_i)$ for $i = 0, \dots, n$. Therefore, if the v_i are all positive, then so are the new weights $\widetilde{v}_i$.

Proposition 4.3. *The barycentric rational curve* (3.11) *with nodes t_i, interpolation points p_i, and weights v_i can be expressed in standard form by first reparameterizing it with φ in* (4.4) *for*

$$\lambda = \frac{v_0 t_n - v_n t_0}{v_0(2t_n - 1) - v_n(2t_0 - 1)} \tag{4.5}$$

and then dividing all weights $\widetilde{v}_i$ by $\widetilde{v}_0$, as long as $\lambda \in (0, 1)$.

Proof. According to Lemma 4.2, the first and the last weight of the reparameterised curve are

$$\widetilde{v}_0 = v_0 \lambda \frac{\varphi(t_0)}{t_0} = \frac{v_0}{\delta(t_0)} \lambda(1 - \lambda) \qquad \text{and} \qquad \widetilde{v}_n = v_n \lambda \frac{\varphi(t_n)}{t_n} = \frac{v_n}{\delta(t_n)} \lambda(1 - \lambda), \tag{4.6}$$

where $\delta(t)$ is again the denominator of $\varphi(t)$. It remains to show that the choice of λ in (4.5) guarantees $\widetilde{v}_0 = \widetilde{v}_n$, which is equivalent to $v_0 \delta(t_n) = v_n \delta(t_0)$ by (4.6), so that both weights are 1 after dividing them by $\widetilde{v}_0$. But, as (4.5) implies

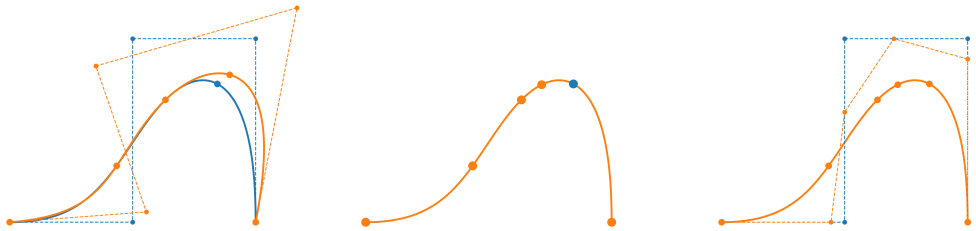
$$\lambda v_0(2t_n - 1) - \lambda v_n(2t_0 - 1) = v_0 t_n - v_n t_0$$

and further

$$v_n t_0 - 2\lambda v_n t_0 + \lambda v_n = v_0 t_n - 2\lambda v_0 t_n + \lambda v_0,$$

we get the desired identity after noting that $\delta(t_0) = \lambda - 2\lambda t_0 + t_0$ and $\delta(t_n) = \lambda - 2\lambda t_n + t_n$. □

The curve cannot be brought into standard form if λ in (4.5) is outside the open interval $(0, 1)$, because φ is singular and no longer a monotonic reparameterization of $[0, 1]$ in that case. However, if $t_0 = 0$, $t_n = 1$, and all v_i are positive, then λ simplifies to $\lambda = v_0/(v_0 + v_n) \in (0, 1)$ and $\widetilde{v}_0 = \widetilde{v}_n = v_0 v_n/(v_0 + v_n) > 0$.



(a) (b) (c)

Figure 4.2. This illustrates the shape editing capabilities using barycentric form. (a) shows the effect of manipulating an interpolation point. (b) illustrates interpolation point sliding. Initial data are given by $v_0 = v_4 = 1$, $v_1 = 6.71875$, $v_2 = 11.25$, $v_3 = 6.71875$ with uniformly distributed nodes $t_i = i/4$. We slide q_3 to $q(0.6)$. Hence $\widetilde{t}_3 = 0.6$, and $\widetilde{v}_0 = \widetilde{v}_4 = 1$, $\widetilde{v}_1 \approx 8.78$, $\widetilde{v}_2 = 30$, $\widetilde{v}_3 \approx 23.53$. (c) illustrates point insertion. For this we insert $\widetilde{q}_3 = q(0.6)$. The new weights are given by $\widetilde{v}_1 \approx 1.67$, $\widetilde{v}_2 \approx 19.2$, $\widetilde{v}_3 = 112.5$, $\widetilde{v}_4 \approx 137.26$, $\widetilde{v}_4 \approx 44.8$, $\widetilde{v}_5 = 2.5$.

4.3 Shape editing using the barycentric form

Once a rational curve is given in barycentric form (4.1), several new options arise for manipulating the curve by modifying the different parameters of the barycentric form: the nodes t_i, the interpolation points q_i, and the weights v_i.

4.3.1 Interpolation point repositioning

The most direct control over the shape of the curve is given by displacing one of the interpolation points, say q_k, while keeping all other parameters fixed. By the interpolation property of the barycentric form, this will force the curve to pass through the new position of q_k at t_k. Compared to moving a Bézier control point p_k, it should be noted that the basis function $(-1)^k \frac{v_k}{t-t_k} \Big/ \sum_{i=0}^{n} (-1)^i \frac{v_i}{t-t_i}$ that corresponds to q_k is neither non-negative nor as nicely "bell-shaped" as the basis function $w_k B_k^n(t) \Big/ \sum_{i=0}^{n} w_i B_i^n(t)$ that corresponds to p_k. Hence, for large displacements, the shape may change less intuitively as it does in the case of editing the control polygon. However, while the general shape of the curve is more easily controlled with the Bézier control points p_i, changing the interpolation points q_i, combined with the "sliding" procedure outlined in Section 4.3.3 and inserting points with ease (see Section 4.4) provides a useful tool for "micro-editing" the curve shape. For example, it can be used to "snap" the curve to some point q that must be interpolated exactly and the interpolation property guarantees that q remains a point on the curve during subsequent editing operations, as long as

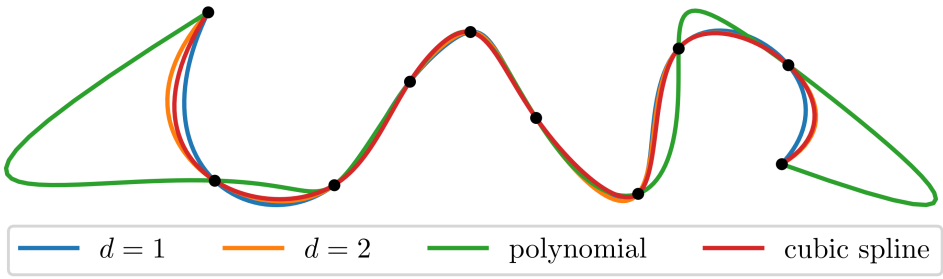


Figure 4.3. This compares the curves that are given by the Floater–Hormann interpolation for $d = 1, 2, n$, and the interpolating cubic-spline for centripetal parametrisation.

q is identical to one of the q_i (see Figure 4.2.a).

4.3.2 Adaptive parametrisation

We can combine several methods to devise an intuitive way to produce the general sketch of the curve using barycentric rational form. First, it is well known that the nodes can be adapted with respect to the control point positions. The most common construction gives

$$\begin{cases} t_0 = 0, \\ t_i = t_{i-1} + \frac{\|P_i - P_{i-1}\|^a}{\sum_{j=1}^{n} \|P_i - P_{i-1}\|^a}, \quad i = 1, \dots, n, \end{cases} \tag{4.7}$$

for $a \in [0, 1]$ (uniform $a = 0$; chord length $a = 1$ [61]; centripetal $a = 0.5$ [73]). There exists also some variants of these parametrisations which take in account for example the angle between consecutive edges [42, 50, 130]. Second, the weights plays a big role in determining the resulting curve. Our experiments show a potential for the use of barycentric interpolation with particular weights in (3.19). We call that we obtain these weights by writing a rational curve as a blend

$$p(t) = \frac{\sum_{i=0}^{n-d} \lambda_i(t) \mathfrak{p}_i(t)}{\sum_{i=0}^{n-d} \lambda_i(t)}, \qquad \lambda_i = \frac{(-1)^i}{(t - t_i) \cdots (t - t_{i+d})},$$

for $d = 0, \dots, n$, and $\mathfrak{p}_i(t)$ is the unique polynomial of degree d that interpolates $p_i, \dots, p_{i+d}$. For $d = 2$, the curve is a blend of $n - 2$ parabolas. This simulates the construction of Catmull–Rom curves where each section is a linear blend of two parabolas [6]. This justify the potential of using the Floater–Hormann interpolation [49] for designing the general shape of curves (see Figure 4.3).

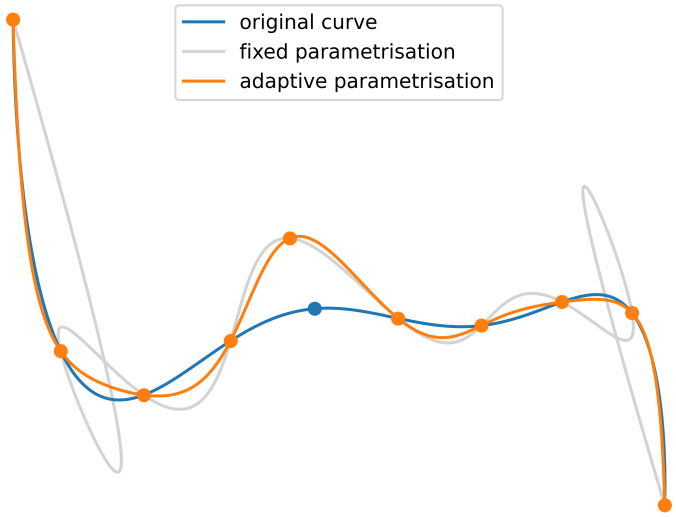


Figure 4.4. This demonstrates the effect of using adaptive reparametrisation.

In fact, the oscillations that we observe when keeping the nodes invariant can be limited by an appropriate combination of the two methods. Consider $\{t_i\}_{i=0,\dots,n}$, $\{\tau_i\}_{i=0,\dots,n}$ be respectively the original set of nodes and the nodes given by adaptive parametrisation (4.7). Similarly, $\{\nu_i\}_{i=0,\dots,n}$, $\{\mu_i\}_{i=0,\dots,n}$ are the original weights and the ones given by (3.19) for $d = 1$ or 2. Suppose we move q_k to a new position $\widetilde{q}_k$. Then by defining some appropriate threshold δ and $s = \min(\|\widetilde{q} - q\|, \delta)$, we set the new nodes and the new weights as

$$\begin{cases} \widetilde{t}_i = t_i \frac{\delta - s}{\delta} + \frac{s}{\delta}\tau_i, \\ \widetilde{\nu}_i = \nu_i \frac{\delta - s}{\delta} + \frac{s}{\delta}\mu_i, \end{cases} \tag{4.8}$$

for $i = 0, \dots, n$. One drawback of this approach is that, while the effect on the curve remains localized (see Figure 4.4), the expected oscillatory behavior is transferred to the rational Bézier control polygon, potentially resulting in a highly irregular and unstructured shape. This, in turn, complicates curve generation via progressive refinement [27] using the curve-splitting technique discussed in Section 2.3.9.

4.3.3 Sliding an interpolation point

Changing one of the nodes, say t_k, in isolation, while keeping the q_i and the ν_i fixed, has a rather unpredictable effect. However, it is possible to preserve

the shape (and the parameterization) of the curve by simultaneously adapting the corresponding q_k and all v_i, so that the effect amounts to "sliding" q_k along the curve (see Figure 4.2.b). This can be achieved by first using Proposition 4.1 to express the curve in Bézier form and then applying (4.2) with the modified nodes to get back to the barycentric form. However, it turns out that we do not have to carry out these conversions explicitly, as the new interpolation points and weights can be expressed directly in terms of the given parameters of the barycentric form.

Proposition 4.4. *Suppose we change the node t_k for some $k \in \{0,\dots,n\}$ to some new value $\overline{t}_k \notin \{t_0,\dots,t_n\}$ and keep the other nodes fixed, that is, we let $\overline{t}_i = t_i$ for $i \neq k$. The barycentric rational curve* (3.11) *with nodes t_i, interpolation points q_i, and weights v_i can then be expressed alternatively in terms of the nodes $\overline{t}_i$, the interpolation points $\overline{q}_k = p(\overline{t}_k)$ and $\overline{q}_i = q_i$ for $i \neq k$, and the weights*

$$\overline{v}_k = \sum_{i=0}^{n} (-1)^{n+k+i} \frac{\overline{t}_k - t_k}{\overline{t}_k - t_i} v_i, \qquad \overline{v}_i = \frac{t_i - t_k}{t_i - \overline{t}_k} v_i, \quad i \neq k. \tag{4.9}$$

Proof. To prove this statement, we stick to the idea sketched out above. After converting the given curve to Bézier form, it follows directly from (4.2) that the new interpolation points are $\overline{q_i} = p(\overline{t}_i)$, which simplifies to $\overline{q}_i = p(t_i) = q_i$ for $i \neq k$. Moreover, we know that the given weights satisfy $v_i = (-1)^{n+i}\omega_i \widehat{z}(t_i)$, where w_i is defined in (3.6) and the denominator polynomial $\widehat{z}$ can be written, independently of the Bézier form, in first barycentric form as $\widehat{z}(t) = \ell(t)\sum_{i=0}^{n}(-1)^i \frac{v_i}{t-t_i}$. Likewise, the new weights satisfy $\overline{v}_i = (-1)^{n+i}\overline{\omega}_i \widehat{z}(\overline{t}_i)$, where $\overline{\omega}_i = \prod_{j=0,\, j\neq i}^{n} \frac{1}{\overline{t}_i - \overline{t}_j}$. If $i \neq k$, then this expression simplifies to

$$\begin{aligned}
\overline{v}_i &= (-1)^{n+i} \frac{1}{\overline{t}_i - \overline{t}_k} \prod_{j=0,\, j\neq i,k}^{n} \frac{1}{\overline{t}_i - \overline{t}_j} \widehat{z}(\overline{t}_i) \\
&= (-1)^{n+i} \frac{1}{t_i - \overline{t}_k} \prod_{j=0,\, j\neq i,k}^{n} \frac{1}{t_i - t_j} \widehat{z}(t_i) \\
&= \frac{t_i - t_k}{t_i - \overline{t}_k} v_i,
\end{aligned}$$

because $\overline{t}_i = t_i$ for $i \neq k$. For the remaining weight $\overline{v}_k$, note that

$\ell(\overline{t}_k) = \prod_{j=0}^{n}(\overline{t}_k - t_j) = (\overline{t}_k - t_k)/\overline{w}_k$, hence

$$\begin{aligned}\overline{v}_k &= (-1)^{n+k}\overline{w}_k\ell(\overline{t}_k)\sum_{i=0}^{n}(-1)^i\frac{v_i}{\overline{t}_k - t_i}\\ &= \sum_{i=0}^{n}(-1)^{n+k+i}\frac{\overline{t}_k - t_k}{\overline{t}_k - t_i}v_i.\end{aligned}$$

□

In an interactive application, this "sliding" of q_k can be realized, for example, by letting the user click on the desired interpolation point, while holding the 'shift' key (to distinguish the action from a displacement of q_k; see below), and translating the subsequent mouse movement (left/right or up/down) into an increase or decrease of t_k until the mouse button is released. Note that the time complexity for updating v_k is $O(n)$, $O(1)$ for updating each of the other v_i, and $O(n)$ for updating q_k [53], hence $O(n)$ overall, which is much more efficient than computing the conversion to Bézier form and back.

For the reasons pointed out in Section 4.1, it seems reasonable to prevent sliding the endpoints q_0 and q_n, that is, to exclude the cases $k = 0$ and $k = n$ in Proposition 4.4, and to restrict $\overline{t}_k$ to the open interval (t_{k-1}, t_{k+1}), so that $\overline{q}_k$ remains between its neighbours q_{k-1} and q_{k+1} along the curve. In this case, it follows immediately from (4.9) that $\operatorname{sign}(\overline{v}_i) = \operatorname{sign}(v_i)$ for $i \neq k$, hence the positivity of the weights v_i carries over to the new weights $\overline{v}_i$. For $\overline{v}_k$, this is not obvious from (4.9), but implied by the fact that changing t_k does not change the curve, so that the non-singularity of the curve still guarantees that all v_i have the same sign [108].

4.3.4 Individual weight change

It remains for us to discuss what happens to the curve if we change one of the weights, say v_k, and it turns out that we can use this parameter to modify the "flatness" of the curve at q_k. Indeed, by (3.21), the tangent vector at q_k is

$$q'(t_k) = \frac{\sum_{i=0,\,i\neq k}^{n}(-1)^{k+i+1}\frac{v_i}{t_k - t_i}(q_k - q_i)}{v_k}.$$

As the numerator, which determines the direction of the tangent at q_k, does not depend on v_k, which in turn appears only in the denominator, it follows that v_k controls the length of the tangent, but not its direction. Therefore, decreasing v_k

"flattens" the curve locally at q_k, while increasing v_k has the effect of letting the curve bend more tightly at q_k.

By saying that v_k affects the local flatness of the curve at q_k, it means that v_k affects the curvature at q_k. In fact, the curvature κ at t_k is given by [5]

$$\kappa = \frac{\det(p'(t_k), p''(t_k))}{\|p'(t_k)\|^3}. \tag{4.10}$$

We recall that from (3.21), we have

$$q''(t_k) = 2\frac{\sum_{i=0, i\neq k}^{n} (-1)^{k+i+1} \frac{v_i}{(t_k - t_i)^2}(q_k - q_i - p'(t_k)(t_k - t_i))}{v_k}.$$

Therefore, by setting

$$U_k = \sum_{i=0, i\neq k}^{n} (-1)^{k+i+1} \frac{v_i}{t_k - t_i}(q_k - q_i),$$
$$V_k = 2\sum_{i=0, i\neq k}^{n} (-1)^{k+i+1} \frac{v_i}{(t_k - t_i)^2}(q_k - q_i),$$

we have

$$\kappa = v_k \frac{\det(U_k, V_k)}{\|U_k\|^3}. \tag{4.11}$$

Now we devise a method to adjust a weight by a visual mean. We know that a curvature can be visually represented by an osculating circle with radius $R(v_k) = 1/\kappa(v_k)$ [5]. However, as these circles can be infinitely large, it might not fit into the user's drawing canvas. Since the most common use of rational curves are to represent conic sections, we are going to use similar manipulation technique as the ρ-values (2.31). In this case, we take

$$\rho_k = \frac{v_k}{v_k + 1}. \tag{4.12}$$

To represent it visually, we consider a point r_k as the intersection the line supported by the normal vector at q_k and the line $(q_{k-1}q_{k+1})$ (see Figure 4.5). We can adjust the weight by dragging a point $v_\star$ such that

$$v_\star = q_k(1 - \rho_k) + \rho_k r_k.$$

Consequently, we have the weight v_k as

$$v_k = \frac{\rho_k}{1 - \rho_k}, \qquad \rho_k = \frac{\|q_k - v\star\|}{\|q_k - r_k\|}.$$

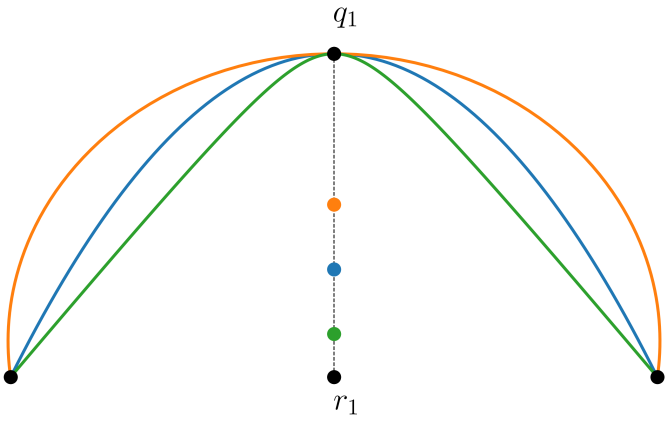


Figure 4.5. This shows the effect of manipulating the weight via the auxiliary point $v_\star$.

Some limits on the possible values of v_k need to be respected, if we want to guarantee that the curve remains non-singular.

Proposition 4.5. *Consider a non-singular barycentric rational curve* (3.11) *with nodes* $0 = t_0 < t_1 < \cdots < t_n = 1$*, interpolation points* q_i*, and weights* $v_i > 0$ *and suppose we change the weight* v_k *for some* $k \in \{0, \dots, n\}$ *to some new value* $\overline{v}_k$*. Then the modified curve* $\overline{P}$ *remains non-singular as long as* $\overline{v}_k \in (M_\star, M^\star)$*, where*

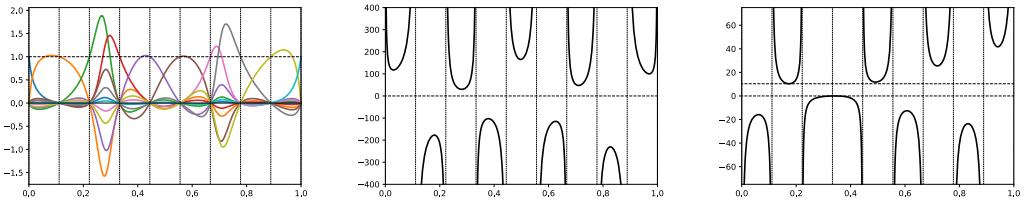
$$M_\star = \max\{\dots, M_{k-2}, M_k, M_{k+2}, \dots\},\ M^\star = \min\{\dots, M_{k-3}, M_{k-1}, M_{k+1}, M_{k+3}, \dots\},$$

with

$$\begin{aligned} M_{k+i} &= \begin{cases} \max\{S_k(t) : t \in (t_{k+i+i_\star}, t_{k+i+i^\star})\}, & i \text{ even}, \\ \min\{S_k(t) : t \in (t_{k+i+i_\star}, t_{k+i+i^\star})\}, & i \text{ odd}, \end{cases} \\ i_\star &= \begin{cases} -1, & i \le 0, \\ 0, & i > 0, \end{cases} \\ i^\star &= \begin{cases} 0, & i < 0, \\ 1, & i \ge 0, \end{cases} \end{aligned} \tag{4.13}$$

and

$$S_k(t) = \sum_{i=0\, i\neq k}^{n} (-1)^{k+i+1} \frac{t - t_k}{t - t_i} v_i.$$



(a) (b) (c)

Figure 4.6. Plots of (a) the basis functions $C_i(t) = (-1)^i \frac{v_i}{t-t_i} \Big/ \sum_{j=0}^{n} (-1)^j \frac{v_j}{t-t_j}$ of a barycentric rational curve of degree $n = 9$ with equidistant nodes $t_i = i/n$ and weights $(v_0, \ldots, v_n) = (1, 8, 3, 2, 5, 6, 2, 5, 8, 1)$, (b) the denominator $D(t)$, and (c) the function $S_k(t)$ for $k = 3$. The horizontal lines represent M_* and M^*.

Proof. First recall that the denominator $D(t) = \sum_{i=0}^{n} (-1)^i \frac{v_i}{t-t_i}$ of a non-singular barycentric rational curve with positive weights v_i (see Figure 4.6.b) satisfies

$$(-1)^n \ell(t) D(t) = \widehat{z}(t) > 0, \qquad t \in [0,1], \tag{4.14}$$

where $\ell(t) = \prod_{i=0}^{n} (t - t_i)$. Next observe that the denominator

$$\overline{D}(t) = (-1)^k \frac{\overline{v}_k}{t - t_k} + \sum_{i=0,\, i \neq k}^{n} (-1)^i \frac{v_i}{t - t_i}$$

of the modified curve $\overline{p}$ vanishes at $\overline{t}$, if and only if $\overline{v}_k = S_k(\overline{t})$. By the interpolation property of barycentric rational curves, it is clear that $\overline{t} \notin \{t_0, \ldots, t_n\}$. Therefore, $\overline{p}$ is non-singular for $t \in [0,1]$, as long as $\overline{v}_k$ is not in the image of $I = [0,1] \setminus \{t_0, \ldots, t_n\}$ under S_k. To better understand the behaviour of S_k, note that

$$S_k(t) = (-1)^{k+1} (t - t_k) D(t) + v_k$$

and assume that $t \in (t_j, t_{j+1})$ for some $j \in \{0, \ldots, n-1\}$. Since $(-1)^{n-j} \ell(t)$ is clearly positive, it follows from (4.14) that $(-1)^j D(t)$ is positive too, with $\lim_{t \to t_j} (-1)^j D(t) = \lim_{t \to t_{j+1}} (-1)^j D(t) = +\infty$. Therefore, $S_k(t) - v_k$ is positive, if $j \geq k$ and $j + k$ is odd, or if $j < k$ and $j + k$ is even, and negative otherwise, converging to $+\infty$ or $-\infty$ as t approaches t_j or t_{j+1}, except at t_k, because $S_k(t_k) = 0$. Consequently, the image of (t_j, t_{j+1}) under S_k is

$$S_k[(t_j, t_{j+1})] = \begin{cases} [M_{j+1}, +\infty), \\ (-\infty, M_{j+1}], \end{cases} \qquad \text{if } j < k-1 \text{ and } j+k \text{ is } \begin{cases} \text{even}, \\ \text{odd}, \end{cases}$$

$$S_k[(t_j, t_{j+1})] = \begin{cases} [M_j, +\infty), \\ (-\infty, M_j], \end{cases} \qquad \text{if } j \geq k+1 \text{ and } j+k \text{ is } \begin{cases} \text{odd}, \\ \text{even}, \end{cases}$$

and

$$S_k[(t_{k-1}, t_{k+1})] = (-\infty, M_k],$$

for the M_j in (4.13) (see Figure 4.6.c). Combining these images, we find that $S_k[I] = (-\infty, M_\star] \cup [M^\star, +\infty)$, which, together with the considerations above, shows that the stated condition for $\overline{v}_k$ guarantees $\overline{P}$ to be non-singular. □

Note that $M_\star$ in Proposition 4.5 is always non-negative, because $S_k(t_k) = 0$ and thus $M_\star \geq M_k \geq 0$, which is in line with our expectation that $\overline{v}_k$ should be positive, just like v_k, in order for $\overline{p}$ to be non-singular. In general, it does not seem feasible to determine the bounds $M_\star$ and $M^\star$ analytically, but they can be computed numerically by first finding the roots of S_k' over the relevant intervals with Newton's method, using, for example, the midpoint of the interval as initial value, and then evaluating S_k at these roots to get the M_j in (4.13).

4.4 Point insertion and degree elevation

A common tool for increasing the flexibility of a rational Bézier curve is degree elevation, which can be used to represent a given curve of degree n as a curve of degree $n+1$ without changing its shape. This increases the number of control points and weights by one and hence gives the user more control to model the desired shape. The equivalent of degree elevation in the barycentric form simply amounts to adding an interpolation point $q_\star = p(t_\star)$ for some $t_\star \in [0,1] \setminus \{t_0, \ldots, t_n\}$, adapting the weights v_i, and computing the appropriate new weight for $q_\star$.

Proposition 4.6. *Let $k \in \{0, \ldots, n+1\}$ and $t_\star \in (t_{k-1}, t_k)$, where $t_{-1} = 0$ and $t_{n+1} = 1$. The barycentric rational curve p of degree n in* (4.1) *with nodes t_i, interpolation points q_i, and weights v_i can then be expressed alternatively as a barycentric rational curve $\breve{p}$ of degree $n+1$ with parameters*

$$\breve{t}_i = \begin{cases} t_i, & i < k, \\ t_\star, & i = k, \\ t_{i-1}, & i > k. \end{cases} \qquad \breve{q}_i = \begin{cases} q_i, & i < k, \\ q_\star = P(t_\star), & i = k, \\ q_{i-1}, & i > k. \end{cases} \qquad \breve{v}_i = \begin{cases} \frac{v_i}{t_\star - t_i}, & i < k, \\ \sum_{i=0}^{n} (-1)^{n+k+i} \frac{v_i}{t_i - t_\star}, & i = k, \\ \frac{v_{i-1}}{t_{i-1} - t_\star}, & i > k. \end{cases} \tag{4.15}$$

Proof. As in the proof of Proposition 4.4, we first use Proposition 4.1 to convert p to Bézier form and then conclude from (4.2) that the given weights satisfy $v_i = (-1)^{n+i} \omega_i \widehat{z}(t_i)$, where ω_i is defined in (3.6) and $\widehat{z}(t) = \ell(t) \sum_{i=0}^{n} (-1)^n \frac{v_i}{t - t_i}$.

Likewise, applying (4.2) to the nodes $\check{t}_0, \dots, \check{t}_{n+1}$, it follows that $\check{q}_i = p(\check{t}_i)$, which simplifies to what is stated in (4.15), and that $\check{v}_i = (-1)^{n+1+i}\check{\omega}_i\widehat{z}(\check{t}_i)$, where $\check{\omega}_i = \prod_{j=0,\, j\neq i}^{n+1} \frac{1}{\check{t}_i - \check{t}_j}$. If $i < k$, then this expression simplifies to

$$\begin{aligned}
\check{v}_i &= (-1)^{n+1+i} \frac{1}{\check{t}_i - \check{t}_k} \prod_{j=0,\, j\neq i,k}^{n+1} \frac{1}{\check{t}_i - \check{t}_j} \widehat{z}(\check{t}_i) \\
&= (-1)^{n+1+i} \frac{1}{t_i - t_\star} \prod_{j=0,\, j\neq i}^{n} \frac{1}{t_i - t_j} \widehat{z}(t_i) \\
&= \frac{v_i}{t_\star - t_i},
\end{aligned}$$

because $\check{t}_i = t_i$ for $i < k$ and $\check{t}_i = t_{i-1}$ for $i > k$, and similarly to

$$\begin{aligned}
\check{v}_i &= (-1)^{n+1+i} \frac{1}{\check{t}_i - \check{t}_k} \prod_{j=0,\, j\neq i,k}^{n+1} \frac{1}{\check{t}_i - \check{t}_j} \widehat{z}(\check{t}_i) \\
&= (-1)^{n+1+i} \frac{1}{t_{i-1} - t_\star} \prod_{j=0,\, j\neq i}^{n} \frac{1}{t_{i-1} - t_j} \widehat{z}(t_{i-1}) \\
&= \frac{v_{i-1}}{t_{i-1} - t_\star},
\end{aligned}$$

if $i > k$. For the remaining weight $\check{v}_k$, note that $\ell(\check{t}_k) = \prod_{j=0}^{n}(\check{t}_k - t_j) = 1/\check{\omega}_k$, hence

$$\check{v}_k = (-1)^{n+1+k}\check{\omega}_k \ell(\check{t}_k) \sum_{i=0}^{n} (-1)^i \frac{v_i}{\check{t}_k - t_i} = \sum_{i=0}^{n} (-1)^{n+k+i} \frac{v_i}{t_i - t_\star}.$$

□

At this point, one may ask: what happens if we convert the barycentric rational curve $\check{p}$ of degree $n+1$ with the parameters in (4.15) to Bézier form? But as $\check{p}$ is just a different representation of the same curve p, its Bézier form must simply be the degree-elevated Bézier form of p. Indeed, since degree elevation does not change the denominator polynomial $\widehat{z}$, this fact can also be observed by applying (4.2) to the degree-elevated Bézier form of p and noticing that this gives the parameters in (4.15) (see Figure 4.2.c).

Chapter 5

Evaluation of rational Bézier curves

In the previous chapter, we only explored the use of barycentric interpolation to manipulate rational curves. Barycentric interpolation is also known for its fast evaluation, as it can be evaluated in linear complexity with respect to the number of interpolated points. This presents an alternative of evaluating rational Bézier curves by converting them to barycentric form, then evaluates this later in linear time.

There exist numerous methods for computing rational Bézier curves, such as adaptations of the classic de Casteljau algorithm for polynomials in the rational case (see Section 2.4.1), or more efficient approaches employing Horner-like schemes (see Section 2.4.2 and Section 2.4.3) or basis conversions (see Section 2.4.4, Section 2.4.5, and Section 5.1). We presented the most commonly used algorithms in Section 2.4 and, for each of them, we describe how it is implemented and provide the pseudocode in Appendix A. This chapter is extracted from our work in [53]. The source code is provided in [92].

5.1 Barycentric algorithm

We propose another alternative to convert the rational Bézier representation to a barycentric rational interpolating form (Algorithm 18 or, for a more optimised version, Algorithm 19). In particular, given a set of interpolation points $Q_0, \ldots, Q_n$ with their respective weights $u_0, \ldots, u_n$ and nodes $t_0, \ldots, t_n$, we write a curve in barycentric form as in (3.11) as

$$P(t) = \frac{\sum_{i=0}^{n} \frac{u_i}{t-t_i} Q_i}{\sum_{i=0}^{n} \frac{u_i}{t-t_i}}. \tag{5.1}$$

The barycentric interpolation points and weights are related to the corresponding Bézier ones as

$$Q_i = P(t_i) \qquad \text{and} \qquad u_i = z(t_i)\omega_i, \qquad i = 0, \dots, n,$$

where $z(t)$ is the z-component of $\widehat{P}(t)$ in (2.35). A common choice of the set of nodes is given by uniformly distributed nodes and Chebyshev nodes. Since the Lagrange weights can easily be computed respectively as in (3.16) and (3.18). The weights turn out to be computed in linear time as

$$u_i = (-1)^i \binom{n}{i} z(t_i).$$

or as

$$u_i = (-1)^i \delta_i z(t_i), \qquad \delta_i = \begin{cases} 1/2, & i = 0 \text{ or } i = n, \\ 1, & i = 1, \dots, n-1. \end{cases}$$

For the sake of efficiency, we propose to compute the values $Q_i = P(t_i)$ by evaluating the rational Bézier curve P at t_i through an adapted version of the rational VS algorithm. Doing so, we can also obtain the values $z(t_i)$ within the same algorithm (Algorithm 16 and 17) as

$$z(t_i) = \sum_{i=0}^{n} x^{n-i} \binom{n}{i} w_i \times \begin{cases} t^n, & t > 1/2, \\ (1-t)^n, & t \leq 1/2, \end{cases} \tag{5.2}$$

for x in (2.37).

5.2 Efficiency analysis

Rational de Casteljau (RDC) - Farin de Casteljau (FDC). We recall that evaluating the numerator and the denominator of a rational Bézier curve as in (2.36) and then dividing the results is equivalent to evaluating the spatial curve (2.35) and applying the central projection on the final result. To that, for the RDC, we need to precompute the points $\widehat{P}_i$ as in Algorithm 2, and evaluate $\widehat{P}(t)$ as in Algorithm 3. The FDC algorithm is a more robust alternative of the RDC algorithm described in (**??**) and implemented optimally in Algorithm 4.

Rational VS (RVS) - Rational Horner-Bézier (RHB). In order to optimise the algorithms and to compute them in linear time, we precompute the factors $\binom{n}{k} w_k P_k$ and $\binom{n}{k} w_k$ as in Algorithm 5, and then we evaluate P as in Algorithm 6 and Algorithm 7, respectively.

Linear time geometric (LTG). Although it is not displayed in Formula 2.39, for numerical reasons, the authors deemed necessary to distinguish the cases $t \in [0, 0.5]$ and $t \in (0.5, 1]$ as in Algorithm 8.

Barycentric algorithm with Chebyshev nodes (CHE) - with uniform nodes (UNI). To get the data of the barycentric form, we can use any of the previously cited algorithms. We choose to adapt the RVS algorithm as in Algorithm 16 to get t_i, Q_i, and $z(t_i)$) simultaneously. We recall that we can get $z(t_i)$ as in (5.2). The data of the barycentric form, such as the interpolation points, the weights, and the nodes, are precomputed in Algorithm 17. We present two ways of evaluating the barycentric form. Algorithm 18 evaluates the barycentric form in (5.1) in the classical way. However, since the distributions of the nodes are symmetric, we can compute $P(t)$ and $P(1-t)$ at the same time. For instance, if we want to get the values of $P(t)$ for $t = k/M, k = 0, \ldots, M$, then the number of flops by using Algorithm 19 is $M(n+1)/2$ less than using Algorithm 18.

Rational Wang–Ball (RWB). The weights and the control points of (2.40) are precomputed in Algorithm 10. Despite its recursive appearance in (2.43)–(2.44), the evaluation of a rational Wang–Ball curve is done in linear time in Algorithm 11.

Rational Bernstein–Fourier (RBF). The algorithm for evaluating $\widehat{P}$ is presented in Algorithm 14. However, for a large number of evaluations, we can also use Algorithm 15 to compute $\widehat{P}(t)$ and $\widehat{P}(1-t)$ in parallel in order to have an optimal runtime. In the algorithms, we assume that the computation of x^n, $x \in \mathbb{R}$, in (5.2) is done with a logarithmic algorithm in the worst case, that is, it involves $2\log_2(n)$ multiplications. The computation of z^n, $z \in \mathbb{C}$, in (2.47) can be done using de Moivre's formula $z^n = r^n(\cos(n\theta) + i\sin(n\theta)$, $\theta = \arg z$ and $r = |z|$. However, since a complex multiplication involves 6 real operations, here we assume that it is $12\log_2(n)$.

We compare the number of floating–point operations in the implementation of each method in Table 5.1. We denote by d the dimension of the space, n the degree of the curve, and let h be the cost of evaluating a trigonometric function $(\cos, \sin, \tan)$.

Table 5.1. Comparison between the number of floating–point operations for the preprocessing (top) and the main algorithm (bottom) for each method.

method	preprocessing
RDC	$d(n+1)$
FDC	0
RVS	$(d+1)(n-1)+2d$
RHB	$(d+1)(n-1)+2d$
LTG	0
CHE	$(n+1)(2dn+d+2n+8+2\log_2(n)+2h)+(d+1)(n-1)+2d$
UNI	$(n+1)(2dn+d+2n+7+2\log_2(n))+(d+1)(n-1)+2d$
RWB (n even)	$2dn^2+4dn-2d+\frac{1}{2}n$
RWB (n odd)	$2dn^2+4dn-2d+\frac{1}{2}n-\frac{1}{2}$
RBF	$O(dn\log n)$

method	add/sub	mult	div	total
RDC	$\frac{1}{2}dn(n+1)+1$	$dn(n+1)$	d	$\frac{3}{2}dn^2+\frac{3}{2}dn+d+1$
FDC	$\frac{1}{2}(d+2)n(n+1)+1$	$\frac{1}{2}(2d+3)n(n+1)$	0	$\frac{3}{2}dn^2+\frac{3}{2}dn+\frac{5}{2}n^2+\frac{5}{2}n+1$
RVS	$(d+1)n+1$	$(d+1)n$	$d+1$	$2dn+d+2n+2$
RHB	$(d+1)(n-1)+d+2$	$2dn+3n$	d	$3dn+d+4n+1$
LTG	$(d+2)n+1$	$2(d+2)n$	$n+1$	$3dn+7n+2$
CHE	$(d+2)(n+1)$	$d(n+1)$	$n+1+d$	$2dn+3d+3n+3$
UNI	$(d+2)(n+1)$	$d(n+1)$	$n+1+d$	$2dn+3d+3n+3$
RWB (n even)	$\frac{3}{2}n(d+1)+1$	$\frac{3}{2}n(2d+2)$	$\frac{3}{2}n$	$\frac{9}{2}dn+6n+1$
RWB (n odd)	$\frac{1}{2}(3n-1)(d+1)+1$	$\frac{1}{2}(3n-1)(2d+2)$	$\frac{1}{2}(3n-1)$	$\frac{9}{2}dn-\frac{3}{2}d+6n-1$
RBF (n even)	$\frac{n}{2}(d+4\log_2 n+2)+d+1$	$\frac{n}{2}(2d+8\log_2 n+2)+d$	d	$6n\log_2(n)+2dn+3d+2n+1$
RBF (n odd)	$\frac{n-1}{2}(d+4\log_2 n+2)+2d+2$	$\frac{n-1}{2}(2d+6\log_2 n+2)+2d+2\log_2 n+1$	d	$8n\log_2(n)+2dn+\frac{7}{2}d+2n-4\log_2 n+1$

5.3 Numerical experiments

We implemented all the methods in C++ and computed the exact value $P(t)$ of the Bézier curve in multiple-precision (1024 bit) floating-point arithmetic with the library *MPFR* [52]. Moreover, we used the *Eigen* module [59] to compute the Inverse Fast Fourier Transform in the Bernstein–Fourier algorithm. The results are obtained using an Ubuntu system on a Dell computer with 8 cores i7-10510U CPU 1.80GHz and 16 GiB of RAM. The codes are compiled with *CMake* compiler optimisation flag -O3.

To compare the efficiency of the different algorithms, we performed a first experiment with respect to the degree n of a rational Bézier curve. This is important because, although cubic curves are more familiar and commonly used, higher-degree curves are particularly interesting for achieving more precision

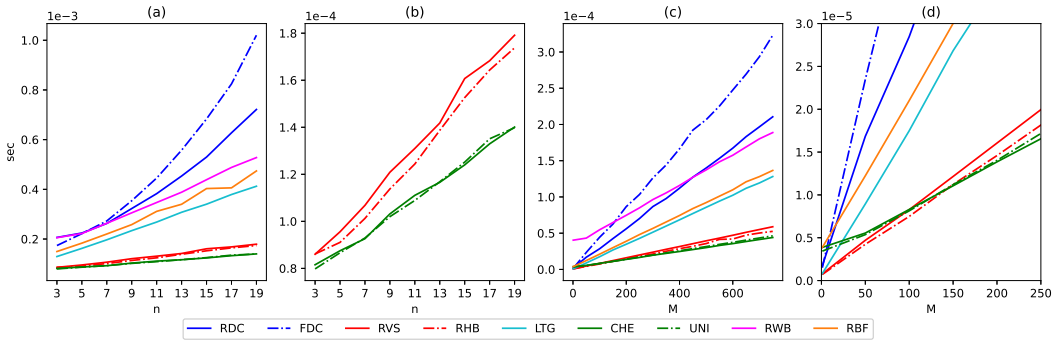


Figure 5.1. Runtime of all algorithms for computing a rational Bézier curve with $P_i = 100i\binom{1}{1} + \binom{1}{1}$ and $w_i = i \mod 2 + 1$. We first consider $M = 2500$ evaluation points for $n = 3, 5, 7, \ldots, 19$ (a) and provide a zoom-in view on the fastest methods (b). Then, we fix the degree at $n = 20$ and vary the number of evaluation points $M = 1, 50, 100, 150, \ldots, 750$ (c), with a zoom-in view on the domain $[1, 250]$ (d).

and smoothness, especially for complex shapes. Therefore, we evaluate rational Bézier curves of degree $n = 3, 5, 7, \ldots, 19$ with control points defined as $P_i = 100i\binom{1}{1} + \binom{1}{1}$ and weights $w_i = i \mod 2 + 1$, $i = 0, \ldots, n$, at $M = 2500$ equidistant evaluation points in $[0, 1]$. The results are obtained by averaging the result of 1000 reruns. Figure 5.1 displays the runtime of all algorithms (a), clearly showing that the RVS, RHB, UNI, and CHE algorithms outperform all the others[1]. It is also evident (b) that the algorithms that use the barycentric form are faster than those employing the Horner-scheme evaluation, with the difference becoming more significant as n increases. This loss of efficiency in the RVS and RHB algorithms is due to the computation of large binomial coefficients with integer arithmetic, which becomes computationally expensive for large values of n.

In the second experiment, we consider the same setup as before, but perform a comparison with respect to the number of evaluation points M. Specifically, we keep the control points and weights consistent with the previous experiment and set $n = 20$ and $M = 1, 50, 100, 150, \ldots, 750$. The results are shown in Figure 5.1 (c) and reconfirm that RVS, RHB, UNI, and CHE are the fastest. However, looking closer in the zoom-in plot (d), we notice that the RVS and RHB algorithms win over the UNI and CHE algorithms only if $M < 200$, otherwise the situation is reversed. This effect arises from the quadratic time preprocessing step required

[1] Bezerra [15] shows that the RBF algorithm is faster than the RVS for $n < 8$, but this does not happen in our experiment, possibly due to a different implementation technique.

by the barycentric algorithms. Although the latter is a one-time operation, it is relevant only for a few evaluations, while it becomes negligible as M grows.

Our analysis and numerical experiments reveal that the fastest algorithms are those employing a Horner-like scheme for the evaluation and those defined in barycentric form. Specifically, while the former is advantageous for scenarios that require the evaluation of the curve at few evaluation points, not exceeding 200, the barycentric form becomes the preferred choice when dealing with a larger number of evaluation points. This is because the preprocessing step required by the barycentric algorithms is executed only once; thus, its runtime becomes negligible for a significant number of evaluations.

Chapter 6

Shape control techniques for periodic rational Bézier curves

In Chapter 4, we analyse the effect of changing the other parameters of the barycentric form, namely the nodes t_i and the weights v_i. The goal of this chapter is to derive similar shape control tools for closed curves, which are described in Section 2.5.

6.1 Equivalence of periodic rational Bézier and trigonometric barycentric form

We recall from Section 3.6 that trigonometric rational curves can be written in barycentric form (3.27) with parameters (3.28). It is important to ask whether the converse is also true.

We recall that rational curves are the image under the projection (2.10) of a spatial curve (2.9). Letting $z_i = z(t_i)$, $\widehat{p}_i = (w_i p_i, w_i)$ and $\widehat{q}_i = (z_i q_i, z_i)$ for $i = 0, \dots, n$, the relations (3.28) can be written compactly as $\widehat{\boldsymbol{Q}} = \boldsymbol{B}\widehat{\boldsymbol{P}}$, where

$$\boldsymbol{B} = \begin{pmatrix} B_0^n(t_0) & \cdots & B_n^n(t_0) \\ \vdots & \ddots & \vdots \\ B_0^n(t_n) & \cdots & B_n^n(t_n) \end{pmatrix}, \qquad \widehat{\boldsymbol{P}} = \begin{pmatrix} \widehat{p}_0 \\ \vdots \\ \widehat{p}_n \end{pmatrix}, \qquad \widehat{\boldsymbol{Q}} = \begin{pmatrix} \widehat{q}_0 \\ \vdots \\ \widehat{q}_n \end{pmatrix}.$$

Proposition 6.1. *We can express the trigonometric barycentric rational curve in* (3.27) *as a periodic rational Bézier curve with control points $p_i = (\widehat{x}_i, \widehat{y}_i)/\widehat{z}_i$ and weights $w_i = \widehat{z}_i$, where $(\widehat{x}_i, \widehat{y}_i, \widehat{z}_i)$ is the i-th row of $\widehat{\boldsymbol{P}} = \boldsymbol{B}^{-1}\widehat{\boldsymbol{Q}}$, as long as all $\widehat{z}_i$ are nonzero.*

Proof. We first use the nodes t_i to compute the ω_i as in (3.26), then use the weights v_i to set $z_i = (-1)^i v_i/\omega_i$, and finally define the i-th row of $\widehat{\boldsymbol{Q}}$ as $(z_i q_i, z_i)$.

The statement then follows from the fact that $\widehat{\boldsymbol{Q}} = \boldsymbol{B}\widehat{\boldsymbol{P}}$, but it remains to show that $\boldsymbol{B}$ is invertible.

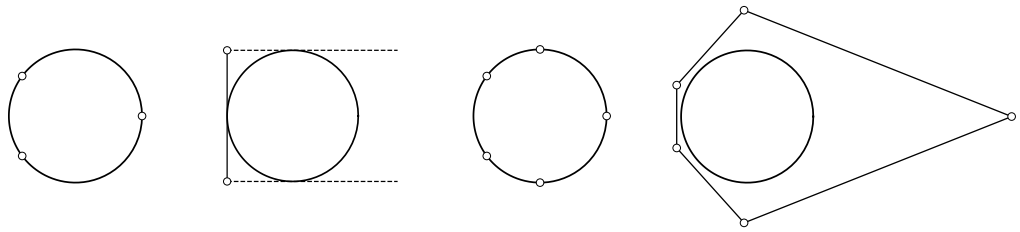
To this end, we first recall that the $n+1$ functions $B_n^i(t) = B_n(t-\phi_i)$, $\phi_i = 2i\pi/(n+1)$, $i = 0,\ldots,n$ span the $(n+1)$-dimensional space

$$T_N = \operatorname{span}\left\{1, \cos(t), \sin(t), \ldots, \cos(Nt), \sin(Nt)\right\}$$

and are thus linearly independent. Moreover, any non-trivial linear combination of these functions is a trigonometric polynomial of order N and, as such, has no more than $2N = n$ zeros [89]. Consequently, the functions $B_i^n(t)$ form a Chebyshev system, which implies that $\boldsymbol{B}$ is non-singular. □

The time complexity of this conversion is $O(n^3)$, since this is the time that it takes in general to solve the dense linear system $\widehat{\boldsymbol{Q}} = \boldsymbol{B}\widehat{\boldsymbol{P}}$. If one or more of the $\widehat{z}_i$ in Proposition 6.1 vanish, then this means that the curve cannot be represented in the periodic rational Bézier form, unless we extend the definition to allow infinite control points, as in the case of classical rational Bézier curves [85, 45, 93]. Alternatively, we can apply degree elevation to represent the curve as a periodic rational Bézier curve with higher degree (see Figure 6.1).

Example 1. Let $N = 1$, so that $n = 2$, and consider the trigonometric barycentric rational curve $q(t)$ with equidistant nodes $t_0 = 0$, $t_1 = 2\pi/3$, $t_2 = 4\pi/3$, interpolation points $q_0 = (2,0)$, $q_1 = (-1,1)$, $q_2 = (-1,-1)$, and weights $v_0 = 2/5$, $v_1 = v_2 = 1$, which turns out to be a non-uniformly parameterized circle with centre $(1/3,0)$ and radius $5/3$ (see Figure 6.1.a). Computing $\widehat{\boldsymbol{P}}$ as in Proposition 6.1, we find that $\widehat{z}_0 = 0$ and that Q is a periodic rational Bézier curve with normal control points $p_1 = (-4/3,5/3)$, $p_2 = (-4/3,-5/3)$ and weights $w_1 = w_2 = 9/10$ and an infinite control point p_0 in the direction $(1,0)$ with homogeneous coordinates $(3/2,0,0)$ (see Figure 6.1.b). After increasing the degree of Q from 1 to 2 by adding two points at $t = \pi/3$ and $t = 5\pi/3$ (see Section 6.3), resulting in the curve $\widetilde{q}$ with nodes $\widetilde{t}_0 = t_0$, $\widetilde{t}_1 = \pi/3$, $\widetilde{t}_2 = t_1$, $\widetilde{t}_3 = t_2$, $\widetilde{t}_4 = 5\pi/3$, interpolation points $\widetilde{q}_0 = q_0$, $\widetilde{q}_1 = q(\widetilde{t}_1) = (1/3,5/3)$, $\widetilde{q}_2 = q_1$, $\widetilde{q}_3 = q_2$, $\widetilde{q}_4 = q(\widetilde{t}_4) = (1/3,-5/3)$, and weights $\widetilde{v}_0 = 8/5$, $\widetilde{v}_1 = \widetilde{v}_4 = 6\sqrt{3}/5$, $\widetilde{v}_2 = \widetilde{v}_3 = 2$ (see Figure 6.1.c), we can apply Proposition 6.1 to convert the curve into periodic rational Bézier form with control points $\widetilde{P}_0 = (7,0)$, $\widetilde{P}_1 \approx (0.254,2.680)$, $\widetilde{P}_2 \approx (-1.444,0.792)$, $\widetilde{P}_3 \approx (-1.444,-0.792)$, $\widetilde{P}_4 \approx (0.254,-2.680)$ and weights $\widetilde{w}_0 = 3/20$, $\widetilde{w}_1 = \widetilde{w}_4 \approx 0.461$, $\widetilde{w}_2 = \widetilde{w}_3 \approx 0.964$ (see Figure 6.1.d). Note that the control polygon is not regular, because of the non-uniform parameterisation of the curve.



(a) (b) (c) (d)

Figure 6.1. Converting a periodic rational curve from trigonometric barycentric form (a) to periodic rational Bézier form may require infinite control points (b), but after inserting two interpolation points (c), which raises the degree of the curve by one, the curve can be represented as a periodic rational Bézier curve with finite control points (d).

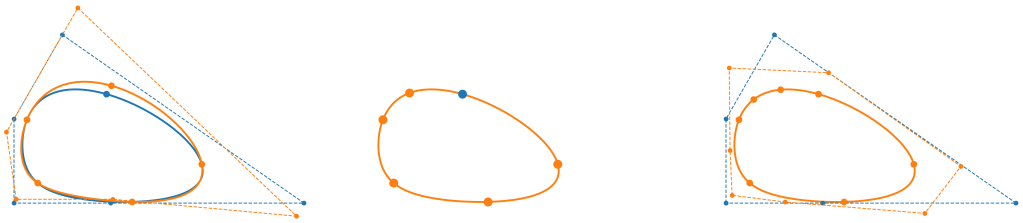
Because of the equivalence of the periodic rational Bézier form (2.8) and the trigonometric barycentric form (3.27), we will use $p(t)$ and $q(t)$ interchangeably from now on to refer to the same periodic rational curve, given in either of the two forms.

6.2 Shape editing using the trigonometric barycentric form

The equivalence of the periodic rational Bézier and the trigonometric barycentric form enables new editing possibilities for periodic rational Bézier curves, beyond changing the control points p_i and the associated weights w_i, by modifying the parameters of the trigonometric barycentric form, namely the interpolation points q_i, the nodes t_i, and the weights v_i, similar to how it can be done for classical rational Bézier curves in Chapter 4.

6.2.1 Displacing an interpolation point

The interpolation property of the trigonometric barycentric form provides an intuitive means to modify the curve by displacing an interpolation point q_k. This is particularly useful for forcing the curve to pass through a specific target point, which is much harder to achieve by changing the Bézier control points p_i. However, in contrast to the basis functions $w_i B_i^n(t)/\sum_{j=0}^n w_j B_i^n(t)$ of the periodic rational Bézier form, the basis functions $(-1)^i \csc\frac{t-t_i}{2} v_i/\sum_{j=0}^n (-1)^j \csc\frac{t-t_j}{2} v_j$ of the trigonometric barycentric form are neither non-negative nor as nicely bell-



(a) (b) (c)

Figure 6.2. This illustrates the shape editing capabilities using barycentric form. (a) shows the effect of manipulating an interpolation point. (b) illustrates interpolation point sliding. Initial data are given by uniform weights $v_i = 1$ and uniformly distributed nodes $t_i = 2i\pi/5$. We slide q_2 to $q(\widetilde{t}_2)$ where $\widetilde{t}_2 = (t_1+t_2)/2)$. It gives $\widetilde{v}_0 = 1$, $\widetilde{v}_1 \approx 1.61$, $\widetilde{v}_2 \approx 1.31$, $\widetilde{v}_3 \approx 0.62$, $\widetilde{v}_4 \approx 0.8$. (c) illustrates point insertion. For this we insert $\widetilde{q}_2 = q(\widetilde{t}_2)$ and $\widetilde{q}_3 = q(\widetilde{t}_3)$ where $\widetilde{t}_2 = (2t_1 + t_2)/3$ and $\widetilde{t}_3 = (t_1+2t_2)/2$. The new weights are given by $\widetilde{v}_0 = \widetilde{v}_5 \approx 4.97$, $\widetilde{v}_1 = \widetilde{v}_2 \approx 88.87$, $\widetilde{v}_3 = \widetilde{v}_4 \approx 37.84$, $\widetilde{v}_6 \approx 3.24$.

shaped and a large displacement of some q_k might therefore induce a less intuitive global deformation of the curve's shape. Hence, while displacing the interpolation points is good for micro-editing the shape (see Figure 6.2.a), the Bézier control points remain the better handles for global shape design. There exists a trigonometric analogue of Floater–Hormann interpolations [74]. This enables effective interpolation point repositioning with adaptive parametrisation.

6.2.2 Sliding an interpolation point

For the same reason as before, changing the value of a single node t_k may deform the curve in a non-intuitive way. However, it is possible to update simultaneously the interpolation point q_k and all weights v_i, such that changing a node t_k results in sliding q_k along a curve (see Figure 6.2.b). This can be achieved in two ways. On the one hand, we can use Proposition 6.1 to express $q(t)$ in periodic rational Bézier form using the current sequence of nodes and then convert $p(t)$ back into trigonometric barycentric form with parameters as in (3.28), this time with the

new sequence of nodes, where t_k is replaced by some new value $\widetilde{t}_k$:

$$\begin{array}{c} q(t) \\ \text{convert to rational Bézier form with respect to } t_0,\dots,t_k,\dots,t_n \big\downarrow \\ p(t) \\ \text{convert to barycentric form with respect to } t_0,\dots,\widetilde{t}_k,\dots,t_n \big\downarrow \\ \widetilde{q}(t) \end{array} \tag{6.1}$$

On the other hand, we can also directly compute the parameters of the new curve $\widetilde{q}(t)$.

Proposition 6.2. *Suppose that we change the node t_k for some $k \in \{1,\dots,n-1\}$ to a new value $\widetilde{t}_k \in (t_{k-1}, t_{k+1})$. The trigonometric barycentric rational curve $q(t)$ in (3.27) can then be expressed in terms of the nodes $t_0,\dots,\widetilde{t}_k,\dots,t_n$, the interpolation points $q_0,\dots,\widetilde{q}_k,\dots q_n$, and the weights $\widetilde{\nu}_i$, where $\widetilde{q}_k = q(\widetilde{t}_k)$ and*

$$\widetilde{\nu}_k = \sin\frac{\widetilde{t}_k - t_k}{2}\sum_{j=0}^{n}(-1)^{j+k}\csc\frac{\widetilde{t}_k - t_j}{2}\nu_j, \qquad \widetilde{\nu}_i = \csc\frac{t_i - \widetilde{t}_k}{2}\sin\frac{t_i - t_k}{2}\nu_i,\ i \neq k. \tag{6.2}$$

Proof. We follow the diagram in (6.1) and first convert $q(t)$ to periodic rational Bézier form (2.8). From (3.28), we then know that the weights of the given curve Q can be written with respect to the given nodes $t_0,\dots,t_n$ as $\nu_i = (-1)^i\omega_i z(t_i)$, where $z(t)$ is the denominator of $p(t)$ and ω_i is defined as in (3.26). It further follows from (3.28) that q can be expressed with respect to the new nodes $\widetilde{t}_0,\dots,\widetilde{t}_n$, where $\widetilde{t}_i = t_i$ for $i \neq k$, using the new interpolation points $\widetilde{q}_i = p(\widetilde{t}_i)$ and the new weights $\widetilde{\nu}_i = (-1)^i\widetilde{\omega}_i z(\widetilde{t}_i)$, where $\widetilde{\omega}_i = \prod_{j=0,\,j\neq i}^{n}\csc\frac{\widetilde{t}_i - \widetilde{t}_j}{2}$. Clearly, $\widetilde{q}_i = p(t_i) = q_i$ for $i \neq k$ and $\widetilde{q}_k = p(\widetilde{t}_k) = q(\widetilde{t}_k)$. To prove the formulas in (6.2), we recall that $z(t)$ can be expressed in first trigonometric barycentric form as

$$z(t) = \ell(t)\sum_{j=0}^{n}\omega_j\csc\frac{t - t_j}{2}z(t_j) = \ell(t)\sum_{j=0}^{n}(-1)^j\csc\frac{t - t_j}{2}\nu_j,$$

where $\ell(t) = \prod_{j=0}^{n}\sin\frac{t-t_j}{2}$. Since $\ell(\widetilde{t}_k) = \prod_{j=0}^{n}\sin\frac{\widetilde{t}_k - t_j}{2} = \sin\frac{\widetilde{t}_k - t_k}{2}/\widetilde{\omega}_k$, we conclude that

$$\begin{aligned}\widetilde{\nu}_k &= (-1)^k\widetilde{\omega}_k z(\widetilde{t}_k)\\ &= (-1)^k\widetilde{\omega}_k\ell(\widetilde{t}_k)\sum_{j=0}^{n}(-1)^j\csc\frac{\widetilde{t}_k - t_j}{2}\nu_j\\ &= \sin\frac{\widetilde{t}_k - t_k}{2}\sum_{j=0}^{n}(-1)^{j+k}\csc\frac{\widetilde{t}_k - t_j}{2}\nu_j.\end{aligned}$$

For $i \neq k$, we note that

$$\begin{aligned}\widetilde{\omega_i} &= \prod_{j=0,\, j\neq i}^{n} \csc\frac{\widetilde{t}_i - \widetilde{t}_j}{2} \\ &= \csc\frac{t_i - \widetilde{t}_k}{2} \prod_{j=0,\, j\neq i,k}^{n} \csc\frac{t_i - t_j}{2} \\ &= \csc\frac{t_i - \widetilde{t}_k}{2} \sin\frac{t_i - t_k}{2}\omega_i,\end{aligned}$$

hence

$$\begin{aligned}\widetilde{v}_i &= (-1)^i \widetilde{\omega}_i z(\widetilde{t}_i) \\ &= \csc\frac{t_i - \widetilde{t}_k}{2} \sin\frac{t_i - t_k}{2}(-1)^i \omega_i z(t_i) \\ &= \csc\frac{t_i - \widetilde{t}_k}{2} \sin\frac{t_i - t_k}{2} v_i.\end{aligned}$$

□

Remark 1. We restrict the new value $\widetilde{t}_k$ to be in the interval (t_{k-1}, t_{k+1}) in Proposition 6.2 and exclude the cases $k = 0$ and $k = n$, so as to keep the statement and the proof simple, but we can also deal with the case $\widetilde{t}_k \in (t_{l-1}, t_l)$ if $0 < l < k$ or $k+1 < l \leq n$. We just need to make sure that the indices are rearranged such that the new nodes are ordered, and this rearrangement requires to change the signs of some weights, so that they all end up having the same sign. For example, if $0 < l < k$, then we can compute the weights $\widetilde{v}_i$ as in (6.2) and then define the correctly ordered nodes $\widehat{t}_i$, the interpolation points $\widehat{q}_i$, and the weights $\widehat{v}_i$ as

i	0	$\dots$	$l-1$	l	$l+1$	$\dots$	k	$k+1$	$\dots$	n
$\widehat{t}_i$	t_0	$\dots$	t_{l-1}	$\widetilde{t}_k$	t_l	$\dots$	t_{k-1}	t_{k+1}	$\dots$	t_n
$\widehat{q}_i$	q_0	$\dots$	q_{l-1}	$\widetilde{q}_k$	q_l	$\dots$	q_{k-1}	q_{k+1}	$\dots$	q_n
$\widehat{v}_i$	$\widetilde{v}_0$	$\dots$	$\widetilde{v}_{l-1}$	$(-1)^{k+l}\widetilde{v}_k$	$-\widetilde{v}_l$	$\dots$	$-\widetilde{v}_{k-1}$	$\widetilde{v}_{k+1}$	$\dots$	$\widetilde{v}_n$

A similar rearrangement table can be derived if $k+1 < l \leq n$ and also for the cases $\widetilde{t}_k \in [0, t_0)$ and $\widetilde{t}_k \in (t_n, 2\pi)$.

It is then natural to ask what happens when $\widetilde{t}_k$ jumps from one interval to the neighbouring interval, for example, as $\widetilde{t}_k$ transitions from (t_{k-1}, t_{k+1}) to (t_{k+1}, t_{k+2}). This can be seen as an elastic collision where q_k slides towards q_{k+1}, swaps the order in the moment of collision, and afterwards continues to slide as q_{k+1}. As long as $h = t_{k+1} - \widetilde{t}_k > 0$, Proposition 6.2 can be used to update

all parameters, but it follows from (6.2) that $\widetilde{v}_k$ and $\widetilde{v}_{k+1}$ grow like $O(1/h)$ as h converges to zero. If h is very small, this can lead to numerical instabilities. However, this can be avoided in an application by limiting the maximal zoom factor and not letting the user move q_k closer than one pixel towards q_{k+1}. In the moment of collision, when $\widetilde{t}_k = t_{k+1}$, the trigonometric barycentric form breaks down as the weights $\widetilde{v}_k$ and $\widetilde{v}_{k+1}$ in (6.2) diverge, but we can still derive a formula for the curve q in this state.

Proposition 6.3. *If the node t_k for some $k \in \{0,\dots,n-1\}$ is set to the new value $\widetilde{t}_k = t_{k+1}$, resulting in a double node t_{k+1}, then we can express the trigonometric barycentric rational curve in* (3.27) *as*

$$q(t) = \frac{\displaystyle\sum_{i=0,i\neq k}^{n} (-1)^i \csc\frac{t-t_i}{2}\widehat{v}_i q_i + (-1)^{k+1}\csc\frac{t-t_{k+1}}{2}\cot\frac{t-t_{k+1}}{2}\widehat{v}'_{k+1}\left(q_{k+1} + 2\tan\frac{t-t_{k+1}}{2}q'_{k+1}\right)}{\displaystyle\sum_{i=0,i\neq k}^{n} (-1)^i \csc\frac{t-t_i}{2}\widehat{v}_i + (-1)^{k+1}\csc\frac{t-t_{k+1}}{2}\cot\frac{t-t_{k+1}}{2}\widehat{v}'_{k+1}}, \quad (6.3)$$

where

$$\begin{aligned}
\widehat{v}_i &= \csc\frac{t_i - t_{k+1}}{2}\sin\frac{t_i - t_k}{2}v_i, \quad i \neq k, k+1,\\
\widehat{v}_{k+1} &= \sin\frac{t_{k+1}-t_k}{2}\sum_{i=0,i\neq k+1}^{n}(-1)^{k+1+i}\csc\frac{t_{k+1}-t_i}{2}v_i + \cos\frac{t_{k+1}-t_k}{2}v_{k+1}, \quad (6.4)\\
\widehat{v}'_{k+1} &= \sin\frac{t_{k+1}-t_k}{2}v_{k+1},
\end{aligned}$$

and $q'_{k+1} = q'(t_{k+1})$.

Proof. If $\widetilde{t}_k \in (t_{k-1}, t_{k+1})$, then we know from Proposition 6.2 that $q(t) = N(t)/D(t)$, where

$$N(t) = \sum_{\substack{i=0\\ i\neq k,k+1}}^{n} (-1)^i \csc\frac{t-t_i}{2}\widetilde{v}_i q_i + (-1)^k \csc\frac{t-\widetilde{t}_k}{2}\widetilde{v}_k\widetilde{q}_k + (-1)^{k+1}\csc\frac{t-t_{k+1}}{2}\widetilde{v}_{k+1}q_{k+1}$$

and

$$D(t) = \sum_{\substack{i=0\\ i\neq k,k+1}}^{n} (-1)^i \csc\frac{t-t_i}{2}\widetilde{v}_i + (-1)^k \csc\frac{t-\widetilde{t}_k}{2}\widetilde{v}_k + (-1)^{k+1}\csc\frac{t-t_{k+1}}{2}\widetilde{v}_{k+1},$$

with $\widetilde{q}_k = q(\widetilde{t}_k)$ and the $\widetilde{v}_i$ as in (6.2). Our task now is to find the limit of $N(t)$ and $D(t)$ as $\widetilde{t}_k$ converges to t_{k+1}.

Let us first focus on the denominator $D(t)$. The sum poses no problem, because the $\widetilde{v}_i$ for $i \neq k, k+1$ simply converge to the $\widehat{v}_i$ in (6.4) as $\widetilde{t}_k \to t_{k+1}$, but the remaining two terms need to be analysed more carefully, since $\widetilde{v}_k$ and $\widetilde{v}_{k+1}$ diverge. Using their definition in (6.2), we have

$$\begin{aligned}
&\csc\frac{t-\widetilde{t}_k}{2}\widetilde{v}_k - \csc\frac{t-t_{k+1}}{2}\widetilde{v}_{k+1} \\
&= \csc\frac{t-\widetilde{t}_k}{2}\sin\frac{\widetilde{t}_k-t_k}{2}\sum_{i=0}^{n}(-1)^{k+i}\csc\frac{\widetilde{t}_k-t_i}{2}v_i \\
&\qquad - \csc\frac{t-t_{k+1}}{2}\csc\frac{t_{k+1}-\widetilde{t}_k}{2}\sin\frac{t_{k+1}-t_k}{2}v_{k+1} \\
&= \csc\frac{t-\widetilde{t}_k}{2}\sin\frac{\widetilde{t}_k-t_k}{2}\sum_{i=0,i\neq k+1}^{n}(-1)^{k+i}\csc\frac{\widetilde{t}_k-t_i}{2}v_i \\
&\qquad + \left(\csc\frac{t-\widetilde{t}_k}{2}\sin\frac{\widetilde{t}_k-t_k}{2} - \csc\frac{t-t_{k+1}}{2}\sin\frac{t_{k+1}-t_k}{2}\right)\csc\frac{t_{k+1}-\widetilde{t}_k}{2}v_{k+1}.
\end{aligned}$$

As before, the terms in the sum are uncritical, and for the last term, we can use L'Hôpital's rule to get

$$\begin{aligned}
&\lim_{\widetilde{t}_k\to t_{k+1}}\left[\left(\csc\frac{t-\widetilde{t}_k}{2}\sin\frac{\widetilde{t}_k-t_k}{2} - \csc\frac{t-t_{k+1}}{2}\sin\frac{t_{k+1}-t_k}{2}\right)\csc\frac{t_{k+1}-\widetilde{t}_k}{2}\right] \\
&\qquad = -\csc\frac{t-t_{k+1}}{2}\cos\frac{t_{k+1}-t_k}{2} - \csc\frac{t-t_{k+1}}{2}\cot\frac{t-t_{k+1}}{2}\sin\frac{t_{k+1}-t_k}{2}.
\end{aligned}$$

Overall, we conclude that $D(t)$ converges to the denominator in (6.3) as $\widetilde{t}_k \to t_{k+1}$.

Similar arguments can be used to show that $N(t)$ converges to the numerator in (6.3). We just need to remember that also $\widetilde{q}_k$ depends on $\widetilde{t}_k$ when applying L'Hôpital's rule to

$$\lim_{\widetilde{t}_k\to t_{k+1}}\left[\left(\csc\frac{t-\widetilde{t}_k}{2}\sin\frac{\widetilde{t}_k-t_k}{2}\widetilde{q}_k - \csc\frac{t-t_{k+1}}{2}\sin\frac{t_{k+1}-t_k}{2}q_{k+1}\right)\csc\frac{t_{k+1}-\widetilde{t}_k}{2}\right],$$

which eventually leads to the term involving q'_{k+1}. □

6.2.3 Changing a weight

We now investigate the effect of changing a weight v_k. In the case of barycentric rational curves, Ramanantoanina and Hormann [93] observed that only the length of the curve's tangent vector at q_k depends on v_k, but not its direction, and we observe the same behaviour in the trigonometric setting (see Figure 6.3).

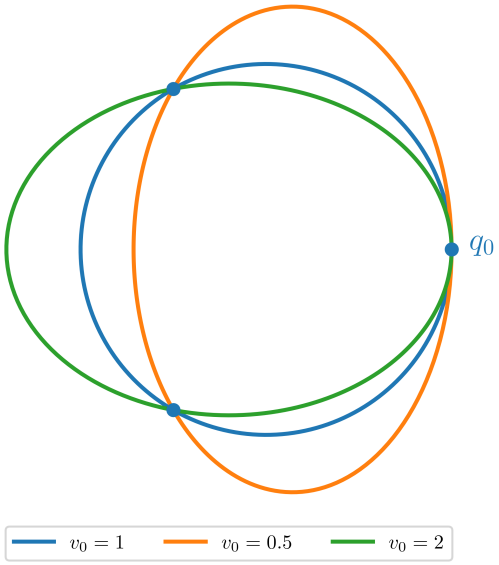


Figure 6.3. This is the result of changing the weight v_0 of q_0. The nodes are given by $t_i = 2i\pi/3$.

Proposition 6.4. *For any $k \in \{0,\dots,n\}$ the tangent vector of the trigonometric rational barycentric curve $q(t)$ in (3.27) at q_k is given by*

$$q'(t_k) = \frac{1}{2v_k} \sum_{i=0,\, i\neq k}^{n} (-1)^{k+i+1} \csc \frac{t_k - t_i}{2} v_i (q_k - q_i). \tag{6.5}$$

Proof. Multiplying the numerator and denominator in (3.27) by $\sin \frac{t-t_k}{2}$, we have $q(t) = N(t)/D(t)$, where

$$N(t) = \sum_{i=0,\, i\neq k}^{n} (-1)^i \sin \frac{t-t_k}{2} \csc \frac{t-t_i}{2} v_i q_i + v_k q_k$$

and

$$D(t) = \sum_{i=0,\, i\neq k}^{n} (-1)^i \sin \frac{t-t_k}{2} \csc \frac{t-t_i}{2} v_i + v_k.$$

By the product rule,

$$N'(t) = \frac{1}{2} \sum_{i=0,\, i\neq k}^{n} (-1)^i \left(\cos \frac{t-t_k}{2} - \sin \frac{t-t_k}{2} \cot \frac{t-t_i}{2} \right) \csc \frac{t-t_i}{2} v_i q_i$$

and

$$D'(t) = \frac{1}{2} \sum_{i=0,\, i\neq k}^{n} (-1)^i \left(\cos\frac{t-t_k}{2} - \sin\frac{t-t_k}{2} \cot\frac{t-t_i}{2} \right) \csc\frac{t-t_i}{2} v_i.$$

For $t = t_k$, these expressions simplify to

$$N(t_k) = (-1)^k v_k q_k, \qquad N'(t_k) = \frac{1}{2} \sum_{i=0,\, i\neq k}^{n} (-1)^i \csc\frac{t_k-t_i}{2} v_i q_i,$$

$$D(t_k) = (-1)^k v_k, \qquad D'(t_k) = \frac{1}{2} \sum_{i=0,\, i\neq k}^{n} (-1)^i \csc\frac{t_k-t_i}{2} v_i.$$

The statement then follows from the quotient rule, which asserts that

$$q'(t_k) = \frac{N'(t_k)D(t_k) - N(t_k)D'(t_k)}{D(t_k)^2}.$$

□

Since v_k does not appear in the sum in (6.5), which determines the direction of the tangent vector at q_k, but only in the denominator of the leading factor, which influences the length of this vector, it follows that decreasing v_k increases the flatness of the curve at q_k, while increasing v_k lets the curve bend more tightly. However, setting v_k to a very small or very large value can create poles. Bounds on v_k that prevent this can be derived in the same way as for barycentric rational curves [93, Proposition 6].

6.3 Degree elevation

Degree elevation of periodic rational Bézier curves are discussed in 2.5.2. We explore two additional variants that we can use.

6.3.1 Degree elevation via the trigonometric barycentric form

The equivalence of the periodic rational Bézier and the trigonometric barycentric form offers an alternative approach to degree elevation. Given a periodic rational Bézier curve $p(t)$ of degree N, we first convert P to a barycentric form (3.28), but using $n+3$ nodes $0 \le t_0 < \cdots < t_{n+2} < 2\pi$, thus resulting in a trigonometric

barycentric rational curve $q(t)$ of degree $N+1$. We then follow Proposition 6.1 to convert Q back into periodic rational Bézier form, but now with degree $N+1$:

$$\begin{array}{rc} & p_N(t) \\ \text{convert to barycentric form with respect to } t_0,\ldots,t_{n+2} & \big\downarrow \\ & q_{N+1}(t) \\ \text{convert to rational Bézier form with respect to } t_0,\ldots,t_{n+2} & \big\downarrow \\ & p_{N+1}(t) \end{array}$$

This procedure can be further simplified by using uniformly distributed nodes t_i and recalling the matrix notation of the conversion process. It then follows that we can compute the control points of the degree-raised spatial periodic Bézier curve from the control points of the given spatial periodic Bézier curve as $\widehat{\boldsymbol{P}}_{N+1} = \boldsymbol{C}^{-1}\boldsymbol{D}\widehat{\boldsymbol{P}}_N$, where the entries of the matrices $\boldsymbol{C} \in \mathbb{R}^{(n+3)\times(n+3)}$ and $\boldsymbol{D} \in \mathbb{R}^{(n+3)\times(n+1)}$ are

$$\boldsymbol{C}_{i,j} = B_{n+2}(\psi_i - \psi_j), \qquad \boldsymbol{D}_{j,k} = B_n(\psi_j - \phi_k), \qquad i,j = 0,\ldots,n+2, \quad k = 0,\ldots,n,$$

with $\psi_i = \frac{2i\pi}{n+3}$, $i = 0,\ldots,n+2$ and $\phi_i = \frac{2i\pi}{n+1}$, $i = 0,\ldots,n$. Hence, the degree can be raised from N to $N+1$ (see Figure 2.13) in three simple steps:

$$\begin{array}{ccc} p_N(t) & & p_{N+1}(t) \\ {\scriptstyle \Pi^{-1}}\big\downarrow & & \big\uparrow{\scriptstyle \Pi} \\ \widehat{p}_N(t) & \xrightarrow[\boldsymbol{C}^{-1}\boldsymbol{D}]{\text{multiply with}} & \widehat{p}_{N+1}(t) \end{array}$$

Instead of inverting $\boldsymbol{C}$, it is advisable to solve instead the linear system $\boldsymbol{C}\widehat{\boldsymbol{P}}_{N+1} = \boldsymbol{D}\widehat{\boldsymbol{P}}_N$, which can be done efficiently in $O(n\log n)$ time using the fast Fourier transform, because $\boldsymbol{C}$ is a symmetric circulant matrix.

6.3.2 Degree elevation using point insertion

If we prefer to work with the trigonometric barycentric form, then a third variant of degree elevation is the following. Given the trigonometric barycentric rational curve $q(t)$, its degree can be raised from N to $N+1$ by inserting two new interpolation points. Conceptually, this is achieved by first converting Q with Proposition 6.1 to periodic rational Bézier form, using the given nodes $t_0,\ldots,t_n$. After that, with the new nodes $\widetilde{t}_0,\ldots,\widetilde{t}_{n+2}$, which are obtained by adding the two

parameter values, we obtain Q as a trigonometric barycentric rational curve of degree $N+1$ with parameters (3.28):

$$\begin{array}{r} q_N(t) \\ \text{convert to rational Bézier form with respect to } t_0,\dots,t_n \big\downarrow \\ p_N(t) \\ \text{convert to barycentric form with respect to } \widetilde{t}_0,\dots,\widetilde{t}_{n+2} \big\downarrow \\ q_{N+1}(t) \end{array}$$

As in the previous subsection, the interpolation points and weights of the degree-raised curve can be computed from the interpolation points and weights of the given curve as $\widehat{\boldsymbol{Q}}_{N+1} = \boldsymbol{C}\boldsymbol{D}^{-1}\widehat{\boldsymbol{Q}}_N$, where the entries of the matrices $\boldsymbol{C} \in \mathbb{R}^{(n+3)\times(n+1)}$ and $\boldsymbol{D} \in \mathbb{R}^{(n+1)\times(n+1)}$ are

$$\boldsymbol{C}_{i,j} = B_n(\widetilde{t}_i - \phi_j), \qquad \boldsymbol{D}_{j,k} = B_n(t_j - \phi_k), \qquad i = 0,\dots,n+2, \quad j,k = 0,\dots,n,$$

but we can actually avoid matrix multiplication and inversion and compute the parameters of the degree-raised curve directly with simple formulas.

Since we can slide interpolation points to any position along the curve (see Proposition 6.2 and Remark 1), we assume without loss of generality that the two new interpolation points are inserted at the two parameter values $\widetilde{t}_k, \widetilde{t}_{k+1} \in (t_{k-1}, t_k)$ for some $k \in \{1,\dots,n\}$ (see Figure 6.2.c), so that the new nodes $\widetilde{t}_0,\dots,\widetilde{t}_{n+2}$ are

i	0	…	$k-1$	k	$k+1$	$k+2$	…	$n+2$
$\widetilde{t}_i$	t_0	…	t_{k-1}	$\widetilde{t}_k$	$\widetilde{t}_{k+1}$	t_k	…	t_n

Proposition 6.5. *The trigonometric barycentric rational curve Q of degree N in* (9.6) *can be expressed as a trigonometric barycentric rational curve $\widetilde{Q}$ of degree $N+1$ with parameters*

$$\widetilde{t}_i = \begin{cases} t_i, \\ \widetilde{t}_k, \\ \widetilde{t}_{k+1}, \\ t_{i-2}, \end{cases} \qquad \widetilde{q}_i = \begin{cases} q_i, \\ q(\widetilde{t}_k), \\ q(\widetilde{t}_{k+1}), \\ q_{i-2}, \end{cases} \qquad \widetilde{v}_i = \begin{cases} \csc\frac{t_i-\widetilde{t}_k}{2}\csc\frac{t_i-\widetilde{t}_{k+1}}{2}\,v_i, \\ \csc\frac{\widetilde{t}_k-\widetilde{t}_{k+1}}{2}\sum_{j=0}^{n}(-1)^{j+k}\csc\frac{\widetilde{t}_k-t_j}{2}\,v_j, \\ \csc\frac{\widetilde{t}_{k+1}-\widetilde{t}_k}{2}\sum_{j=0}^{n}(-1)^{j+k+1}\csc\frac{\widetilde{t}_{k+1}-t_j}{2}\,v_j, \\ \csc\frac{t_{i-2}-\widetilde{t}_k}{2}\csc\frac{t_{i-2}-\widetilde{t}_{k+1}}{2}\,v_{i-2} \end{cases} \quad \text{if} \quad \begin{cases} i < k, \\ i = k, \\ i = k+1, \\ i > k+1. \end{cases}$$

Proof. From (3.28), the statement is obvious for the nodes $\widetilde{t}_i$ and the interpolation points $\widetilde{q}_i$, and that $v_i = (-1)^i\omega_i z(t_i)$ and $\widetilde{v}_i = (-1)^i\widetilde{\omega}_i z(\widetilde{t}_i)$. For $i < k$, we note that

$$\widetilde{\omega}_i = \prod_{j=0,\,j\neq i}^{n+2} \csc\frac{\widetilde{t}_i-\widetilde{t}_j}{2} = \csc\frac{t_i-\widetilde{t}_k}{2}\csc\frac{t_i-\widetilde{t}_{k+1}}{2}\,\omega_i,$$

and therefore

$$\begin{aligned}\widetilde{v}_i &= (-1)^i \widetilde{\omega}_i z(\widetilde{t}_i) \\ &= \csc\frac{t_i-\widetilde{t}_k}{2}\csc\frac{t_i-\widetilde{t}_{k+1}}{2}(-1)^i\omega_i z(t_i) \\ &= \csc\frac{t_i-\widetilde{t}_k}{2}\csc\frac{t_i-\widetilde{t}_{k+1}}{2}v_i.\end{aligned}$$

For $i > k+1$, we can similarly show that

$$\widetilde{v}_i = \csc\frac{t_{i-2}-\widetilde{t}_k}{2}\csc\frac{t_{i-2}-\widetilde{t}_{k+1}}{2}v_{i-2}.$$

For $i = k$, recall that $z(t) = \ell(t)\sum_{j=0}^{n}(-1)^j \csc\frac{t-t_j}{2}v_j$ and since $\ell(\widetilde{t}_k) = \csc\frac{\widetilde{t}_k-\widetilde{t}_{k+1}}{2}/\widetilde{\omega}_k$, we have

$$\begin{aligned}\widetilde{v}_k &= (-1)^k\widetilde{\omega}_k z(\widetilde{t}_k) \\ &= (-1)^k\widetilde{\omega}_k\ell(\widetilde{t}_k)\sum_{j=0}^{n}(-1)^j\csc\frac{\widetilde{t}_k-t_j}{2}v_j \\ &= \csc\frac{\widetilde{t}_k-\widetilde{t}_{k+1}}{2}\sum_{j=0}^{n}(-1)^{j+k}\csc\frac{\widetilde{t}_k-t_j}{2}v_j.\end{aligned}$$

For $i = k+1$, a similar reasoning gives

$$\widetilde{v}_{k+1} = \csc\frac{\widetilde{t}_{k+1}-\widetilde{t}_k}{2}\sum_{j=0}^{n}(-1)^{j+k+1}\csc\frac{\widetilde{t}_{k+1}-t_j}{2}v_j.$$

□

Remark 2. In practice, a user would probably prefer to insert one interpolation point at a time. Assuming that the current number of interpolation points is odd, this can be done by simply inserting two new points $\widetilde{q}_k$ and $\widetilde{q}_{k+1}$, following Proposition 6.5, but showing only $\widetilde{q}_k$ to the user. Once the user decides to insert another interpolation point, $\widetilde{q}_{k+1}$ is revealed and moved to the desired position using Proposition 6.2.

Chapter 7

Trigonometric tangent interpolating curve

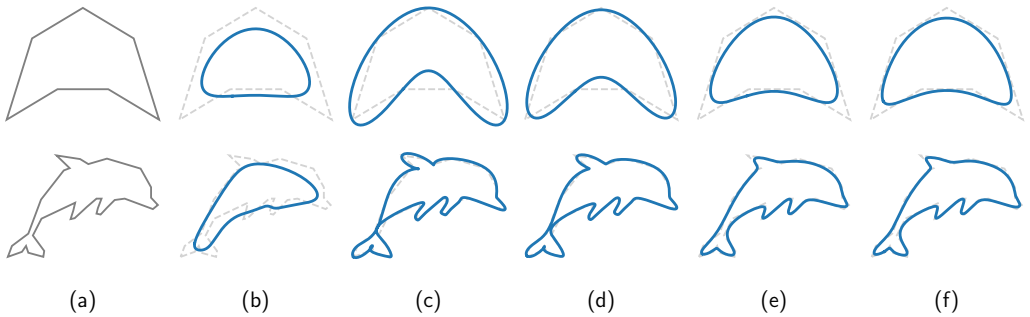


Figure 7.1. Comparison of curves defined by a common closed control polygon (a): while periodic Bézier curves suffer from shrinking as the number of control points increases (b), trigonometric vertex interpolating curves tend to have shape artefacts in the presence of control edges with non-uniform length. Our new trigonometric tangent interpolating curves (of type 1) are always close to the control polygon (d), at the expense of not necessarily being inside the convex hull. We further introduce a variant of these curves which behaves better in that respect. These type-2 curves (e) are very similar to uniform cubic B-spline curves (f).

The periodic Bézier curves in (2.49) share some key properties with the polynomial Bézier curves (2.4). In particular, the curve $p(t)$ is contained in the convex hull of its control points and its shape roughly imitates the shape of the control polygon. The kernel B_n in (2.51) is designed to be somewhat similar to Bernstein polynomials, which are used to express classical Bézier curves. They are non-negative, linearly independent, and form a partition of unity. Moreover, they are

bell-shaped and have a contact of order N with the x-axis (see Figure 7.3). Consequently, designing a periodic Bézier curve is similar to modelling polynomial Bézier curves and very intuitive, especially for low-degree curves. However, as n grows, these basis functions become more and more global and cause the curve to increasingly deviate from the shape of its control polygon, thus smoothing out its details (see Figure 7.1b). Another option is to use an interpolating basis. However, due to the oscillatory nature of the basis functions (see Figure 7.3), this representation is suitable only for small displacements of the control points p_i [94]. For open curves, in Section 2.3.10 we use the Gauss–Legendre control polygon to manipulate a polynomial curve.

We explore two novel alternative representations of trigonometric curves, both based on the interpolation of tangents instead of points (Section 7.1). The first construction gives curves that are very tight to the control polygon (see Figure 7.1d). The second construction is a minor variation of the first, but turns out to give curves (see Figure 7.1e) that are very similar to cubic B-spline curves (see Figure 7.1f), which are the de facto standard for closed curve design. The main advantage of both representations is that they provide more intuitive control over the shape of the curve compared to the two existing approaches for modelling trigonometric curves. This chapter is a recollection of our work in [95].

7.1 Trigonometric tangent interpolating curves

Suppose that we are given a control polygon with $n+1$ control points $p_0, \dots, p_n$, where $n = 2N$ for some $N \in \mathbb{N}$. Our aim is to create two families of curves P_d for $d = 1, 2$, given by trigonometric polynomials in different bases that we denote by $L_i^d(t)$, $i = 0, \dots, n$.

The first family ($d = 1$) is inspired by the construction of polynomial Gauss–Legendre curves [77]. The trigonometric analogue of that approach is the trigonometric polynomial curve P_1 of degree N whose derivative at the *dual* uniformly distributed nodes is parallel to the edge vectors $\Delta_i = p_{i+1} - p_i$ (see Figure 7.2, top), that is,

$$P_1'(\psi_i) = \omega_i \Delta_i, \qquad i = 0, \dots, n, \tag{7.1}$$

with $\psi_i = (2i+1)\pi/(n+1)$, where $p_{n+1} = p_0$ and $\omega_0, \dots, \omega_n$ are certain *weights*, whose values will be determined below. We call this curve a *trigonometric tangent interpolating curve of type 1*.

While the constraints in (7.1) are associated with the edge midpoints of the control polygon, we further propose a variant in which the constraints are associated with the control points instead. To this end, we simply average two

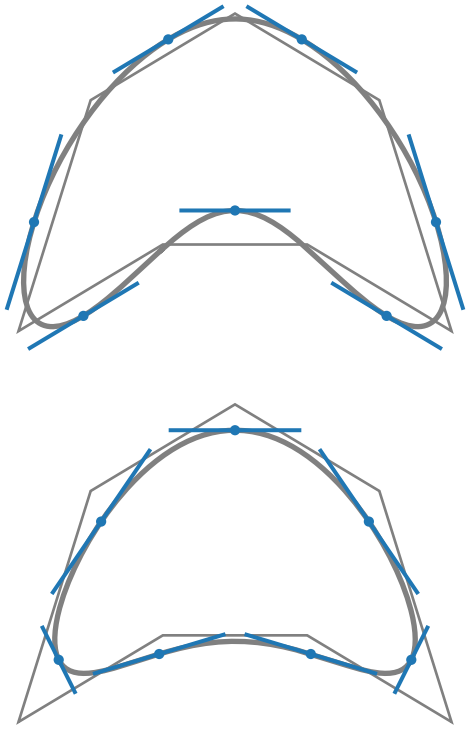


Figure 7.2. Tangent interpolation conditions (7.4) for $n = 6$ at the highlighted points $P_d(t_i)$ for type-1 curves (top) and type-2 curves (bottom).

successive edge vectors and define the *trigonometric tangent interpolating curve of type 2* to be the trigonometric polynomial P_2 whose derivative at the *primal* uniformly distributed nodes is parallel to these averages (see Figure 7.2, bottom), that is,

$$P_2'(\phi_i) = \omega_i \frac{\Delta_{i-1} + \Delta_i}{2}, \qquad i = 0, \dots, n, \tag{7.2}$$

where $p_{-1} = p_n$ and $p_{n+1} = p_0$.

In order to derive the appropriate basis functions L_i^d that allow us to express the type-1 curve P_1 and the type-2 curve P_2 in terms of the control points p_i as

$$P_d(t) = \sum_{i=0}^{n} L_i^d(t) p_i, \tag{7.3}$$

note that we can write the interpolation conditions in (7.1) and (7.2) conve-

niently in a common form as

$$P_d'(t_i) = \omega_i \frac{p_{i+1} - p_{i+1-d}}{d}, \qquad i = 0, \dots, n, \tag{7.4}$$

for $d \in \{1, 2\}$, where $t_i = (2 + 2i - d)\pi/(n+1)$.

For the remainder of the chapter, we assume $(\phi_i)_{i=0,\dots,n}$ to be the uniformly distributed nodes $\phi_i = 2i\pi/(n+1)$ and $(t_i)_{i=0,\dots,n}$ to be a shifted version $t_i = \phi_i - \phi_d/2$ for $i = 0, \dots, n$. We start the construction of L_i^d by recalling that $P_d'(t)$ is a trigonometric polynomial of degree N (with vanishing constant coefficient) and can hence be expressed, using the tangent interpolation conditions (7.4) and the basis functions in (3.24), as

$$P_d'(t) = \sum_{i=0}^{n} \ell_i^n(t) \omega_i \frac{p_{i+1} - p_{i+1-d}}{d}. \tag{7.5}$$

Integrating both sides of (7.5), we get P_d as in (7.3) with basis functions

$$L_i^d(t) = \frac{1}{d}\big(\omega_{i-1} I_{i-1}(t) - \omega_{i-1+d} I_{i-1+d}(t)\big) \tag{7.6}$$

for $i = 0, \dots, n$, where

$$I_i(t) = \int_{t_i}^{t} \ell_i^n(x)\,\mathrm{d}x \tag{7.7}$$

and $\ell_{-1}^n = \ell_n^n$, $\ell_{n+1}^n = \ell_0^n$ and $\omega_{-1} = \omega_n$, $\omega_{n+1} = \omega_0$.

To obtain an explicit expression for $L_i^d(t)$, we recall that the nodes t_i are uniformly spaced with $t_i = t_0 + \phi_i$. Therefore, ℓ_i^n in (3.5) becomes

$$\ell_i^n(t) = K(t - t_i), \tag{7.8}$$

where

$$K(t) = \frac{1}{n+1} \sum_{k=-N}^{N} e^{\mathrm{i}kt} = \frac{1}{n+1}\left(1 + 2\sum_{k=1}^{N} \cos(kt)\right).$$

Indeed, since $e^{\mathrm{i}\phi_j}$ is an $(n+1)$-th root of unity, we find that

$$K(\phi_j) = \frac{1}{n+1} \sum_{k=-N}^{N} e^{\mathrm{i}k\phi_j} = \frac{1}{n+1} \frac{e^{\mathrm{i}(n+1)\phi_j} - 1}{e^{\mathrm{i}\phi_j} - 1} = \delta_{0,j}$$

and

$$K(t_j - t_i) = K(\phi_j - \phi_i) = K(\phi_{j-i}) = \delta_{0,j-i} = \delta_{i,j}$$

Therefore, since $\ell_i^n(t)$ and $K(t-t_i)$ are both trigonometric polynomials of degree N, which agree at the $2N+1$ knots $t_0,\dots,t_n$, they must be identical [89]. We can now use (7.8) to write $I_i(t)$ in (7.7) as

$$I_i(t) = \int_0^{t-t_i} K(x)\,\mathrm{d}x = \frac{1}{n+1}\Big(t - t_i + \sum_{k=1}^{N} \frac{2}{k}\sin\big(k(t-t_i)\big)\Big), \tag{7.9}$$

which in turn can be used in (7.6) to get an explicit formula for L_i^d.

7.1.1 Partition of unity

We shall now choose the weights ω_i such that the basis functions $L_0^d,\dots,L_n^d$ form a partition of unity. Due to the uniformity of the nodes t_i, the only choice that gives the expected symmetry is to set all nodes to a common value $\omega = \omega_0 = \cdots = \omega_n$. It then follows from (7.6) and (7.9) that

$$\begin{aligned} L_i^d(t) &= \frac{\omega}{d}\big(I_{i-1}(t) - I_{i-1+d}(t)\big) \\ &= \frac{\omega}{d(n+1)}\Big(\phi_d + \sum_{k=1}^{N}\frac{2}{k}\Big[\sin\big(k(t-t_{i-1})\big) \\ &\qquad\qquad - \sin\big(k(t-t_{i-1+d})\big)\Big]\Big) \end{aligned} \tag{7.10}$$

and consequently

$$\sum_{i=0}^{n} L_i^d(t) = \frac{\omega}{d}\phi_d = \omega\frac{2\pi}{n+1}.$$

Now it is clear that the basis functions form a partition of unity if and only if $\omega = (n+1)/(2\pi)$. Inserting this value into (7.10) and the fact that $\sin\alpha - \sin\beta = 2\cos\frac{\alpha+\beta}{2}\sin\frac{\alpha-\beta}{2}$ [131], we can finally simplify the formula for L_i^d to

$$L_i^d(t) = \frac{1}{n+1} + \frac{2}{d\pi}\sum_{k=1}^{N}\frac{1}{k}\cos\big(k(t-\phi_i)\big)\sin\frac{k\phi_d}{2}. \tag{7.11}$$

7.1.2 Linear independence

To prove the linear independence of the basis functions $L_0^d,\dots,L_n^d$, we recall that the trigonometric monomials, that is, the components of the vector $\boldsymbol{c}_n(t)$, are clearly linearly independent. Hence, it suffices to show that the vector $\boldsymbol{L}_n^d(t) = \big(L_0^d(t),\dots,L_n^d(t)\big)^T$ can be expressed as $\boldsymbol{L}_n^d(t) = \Gamma_{n,d}\boldsymbol{c}_n(t)$ for some non-singular matrix $\Gamma_{n,d}$ and such that $\boldsymbol{c}_n(t) = \big(1,\cos t,\sin t,\dots,\cos(Nt),\sin(Nt)\big)^T \in \mathbb{R}^{n+1}$.

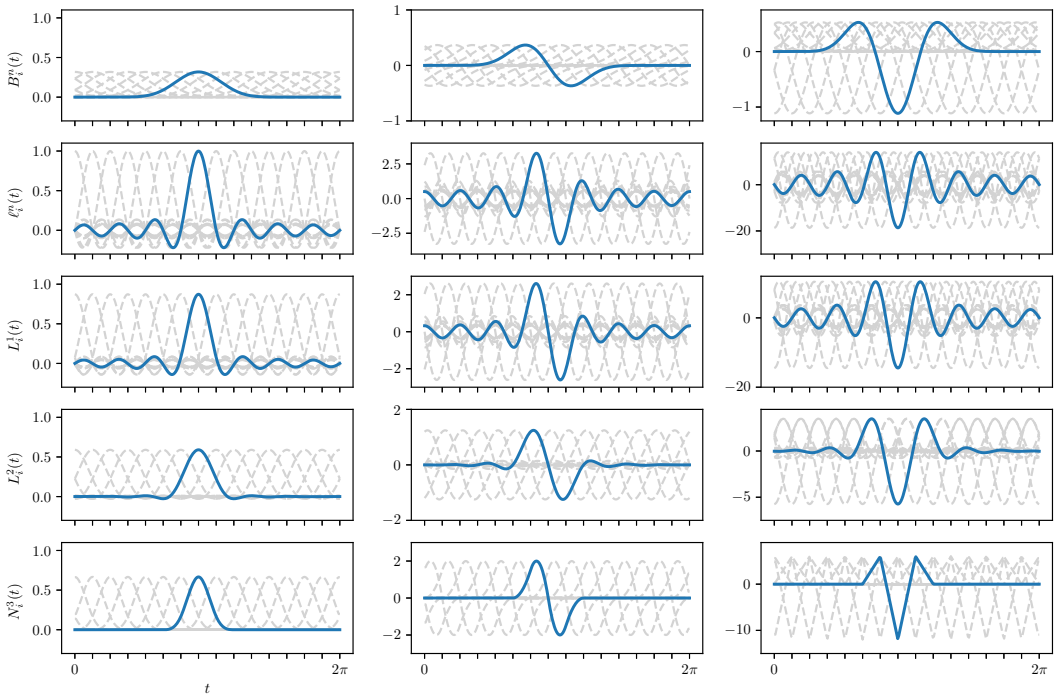


Figure 7.3. A comparison of the different basis functions considered in this paper (left column) for $n = 14$ and the corresponding first derivatives (middle column) and second derivatives (right column). From top to bottom: periodic Bézier, trigonometric Lagrange, trigonometric tangent interpolating type-1 and type-2, and uniform cubic B-splines.

To this end, we expand $L_i^d(t)$ in (7.11) into trigonometric polynomial form,

$$L_i^d(t) = \frac{1}{n+1} + \frac{2}{d\pi}\sum_{k=1}^{N}\frac{1}{k}\cos(kt)\cos(k\phi_i)\sin\frac{k\phi_d}{2} + \frac{2}{d\pi}\sum_{k=1}^{N}\frac{1}{k}\sin(kt)\sin(k\phi_i)\sin\frac{k\phi_d}{2},$$

and conclude that

$$L_i^d(t) = \mathbf{c}_n(\phi_i)^T D_d \mathbf{c}_n(t),$$

where D_d is the diagonal matrix

$$D_d = \mathrm{diag}(a_0, a_1, a_1, \ldots, a_N, a_N),$$

with entries

$$a_0 = \frac{1}{n+1}, \qquad a_i = \frac{2}{id\pi}\sin\frac{i\phi_d}{2}, \quad i = 1, \ldots, N.$$

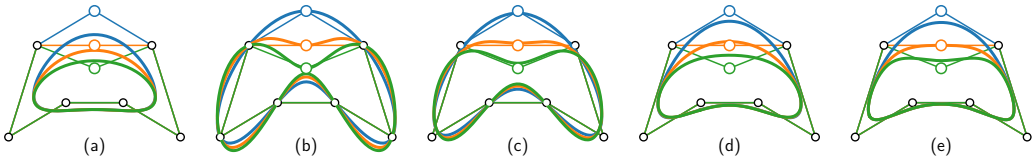


Figure 7.4. Effect of modifying a single control point for various curve types: periodic Bézier curve (a), trigonometric vertex interpolating curve (b), trigonometric tangent interpolating curve of type 1 (c) and type 2 (d), and cubic B-spline curve (e).

Consequently, $\boldsymbol{L}_n^d(t) = \Gamma_{n,d}\boldsymbol{c}_n(t)$, where

$$\Gamma_{n,d} = C_{n,n}D_d, \qquad C_{\mu,\nu} = \big(\boldsymbol{c}_\mu(\phi_0),\ldots,\boldsymbol{c}_\mu(\phi_\nu)\big)^T \in \mathbb{R}^{(\nu+1)\times(\mu+1)},$$

and it remains to show that $\Gamma_{n,d}$ is non-singular, which follows from two observations. On the one hand, we note that $\sin\frac{i\phi_d}{2} \neq 0$ for $i = 1,\ldots,N$. On the other hand, we recall that the functions $1, \cos t, \sin t, \ldots, \cos(Nt), \sin(Nt)$ form a Chebyshev system, that is, they are linearly independent and no non-trivial linear combination of them admits more than n zeros in $[0, 2\pi)$ [89]. Therefore, D_d and $C_{n,n}$ are both non-singular.

7.2 Practical aspects

7.2.1 Implementation

The implementation of our tangent interpolating curves is straightforward. Given some parameter $t \in [0, 2\pi]$, we first evaluate each of the $n+1$ basis functions L_i^d in $O(n)$ time, using (7.11), and then compute the corresponding curve point $P_d(t)$ using (7.3) in $O(n)$ time. Hence, the evaluation of a single point requires $O(n^2)$ time. However, if we render the whole curve with mn equidistant samples (i.e., m samples per interval $[\phi_i, \phi_{i+1}]$), then we can exploit the fact that the basis functions are identical up to uniform shifts and get by with evaluating just one basis function at all sample points in $O(mn^2)$ time and then computing all mn curve points as before, again in $O(mn^2)$ time (see Algorithm 1). Overall, this amounts to an $O(n)$ time complexity per curve point. For example, our simple *Python* implementation handles degree $n = 101$ curves with $m = 10000$ samples per interval in approximately 50 ms.

Algorithm 1 Optimised sampling algorithm

Require: p (control points), m (samples per interval)
Ensure: c (final curve)

```
n ← len(P) − 1
M ← (n + 1) · m
Initialize c as a zero matrix of size M × 2
u ← 2π/M
for k = 0 to M − 1 do
    t ← k · u
    b ← L_n(n, 0, t)
    for i = 0 to n do
        K ← (k − i · m) mod M
        c[K] ← c[K] + b · P[i]
    end for
end for
return c
```

7.2.2 Curve manipulation

Since the basis functions of our type-1 and type-2 curves are neither non-negative nor non-positive (see Figure 7.3), that is, they do not form a blending system [26], the curves are not necessarily contained in the convex hull of the control points (see Figure 7.4c,d). In our experiments, this usually happens when we have a double point, that is, $p_i = p_{i+1}$, or three consecutive collinear points. In addition, since the nodes ψ_i and ϕ_i are distributed uniformly, it is recommended to keep the lengths of the control edges more or less even, especially for our type-1 curves. Figure 7.4 shows the change of the curve induced by the manipulation of a single control point. These changes are always as intuitive as for cubic B-spline curves and stable in the sense that a small displacement of the control point leads to a small deviation of the curve. Sharp corners (more precisely, cusps) can be created by overlapping three or more consecutive control points (see Figure 7.5). However, the cusp point does not coincide with the overlapping control points, as it would in the case of a cubic B-spline.

7.2.3 Curve similarities

The examples in Figures 7.1, 7.4, and 7.8 show that our type-1 curves are quite similar to vertex interpolating curves and that our type-2 curves are akin to uniform cubic B-spline curves. With reference to Figure 7.3, this is not surprising,

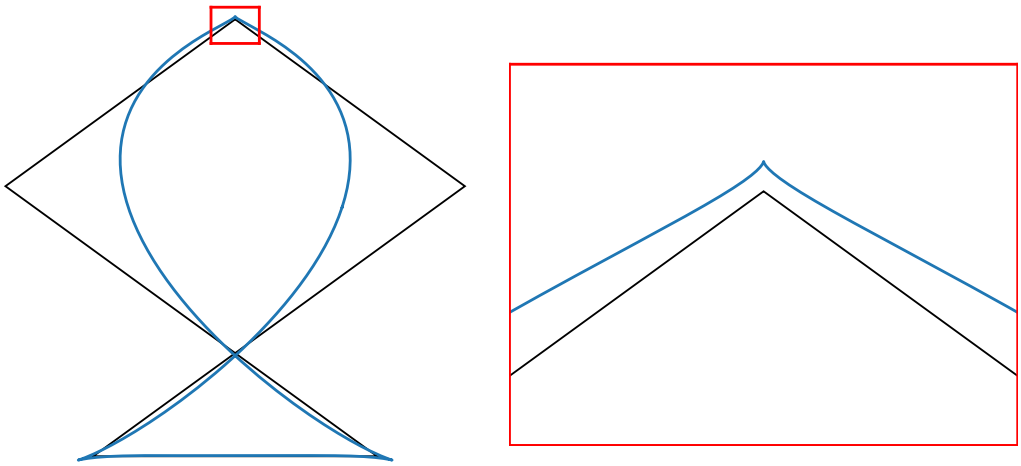


Figure 7.5. An example of a curve given by 11 points with sharp corners (top, bottom left, and bottom right) created by overlapping three successive points with a zoomed-in view of the top corner.

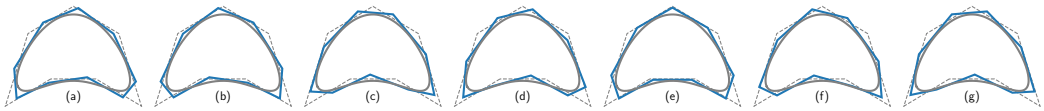


Figure 7.6. Set of all degree-elevated control polygons $\boldsymbol{q}$ (blue) for different choices of p_0. In this example, the optimization (7.12) picks the control polygon (e) as the preferred degree-elevated control polygon, which is the one that keeps the reflection symmetry present in $\boldsymbol{p}$ (dashed).

since the same similarity can be observed for the respective basis functions and their first and second derivatives.

The similarity of our type-2 curves to uniform cubic B-spline curves actually goes beyond mere visual inspection. On the one hand, both curves have tangents that are parallel to $p_{i+1} - p_{i-1}$ at the nodes ϕ_i, because the only non-vanishing basis functions at ϕ_i are those with indices $i-1$ and $i+1$. On the other hand, in both cases, the basis function with index i vanishes at all nodes ϕ_j with indices $j \neq i-1, i+1$.

7.2.4 Degree elevation

Apart from moving the control points, a typical operation in curve modelling is refinement, which increases the number of control points, and thus enables a more detailed manipulation of the curve shape. In the case of our type-1 and type-2 curves, this can be achieved by degree elevation, which increases the number of control points by two. As for classical and periodic Bézier curves, the new control points can be obtained from the old ones via matrix multiplication.

To describe our refinement process, let $m = n+2$ and denote by $\boldsymbol{p}$ and $\boldsymbol{q}$ the (row) vectors $(p_0,\dots,p_n)$ and $(q_0,\dots,q_m)$, respectively. Since $\boldsymbol{L}_i^d(t) = \Gamma_{i,d}\boldsymbol{c}_i(t)$ for $i = n,m$, we can obtain the degree-elevated control polygon $\boldsymbol{q}$ by solving the linear system

$$\boldsymbol{p}\,\Gamma_{n,d}C_{n,m}^T = \boldsymbol{q}\,\Gamma_{m,d}C_{m,m}^T$$

or by computing $\boldsymbol{q}$ directly as $\boldsymbol{q} = \boldsymbol{p}M_{n,m}$, where the matrix

$$M_{n,m} = \Gamma_{n,d}C_{n,m}^T\big(\Gamma_{m,d}C_{m,m}^T\big)^{-1}$$

can be precomputed.

Since we are in a cyclic setting and more focused on shape than parameterisation, we can arbitrarily shift the indices of the control points of $\boldsymbol{p}$ and let any p_i take the role of the starting point p_0. Interestingly, each such shift leads to a different degree-elevated control polygon $\boldsymbol{q}$, but all $n+1$ variants define the same curve. Given this situation, one may wonder which is the best choice? For example, if the original control polygon $\boldsymbol{p}$ has some symmetry, we may prefer $\boldsymbol{q}$ to keep this symmetry.

We propose a simple method to automatically select a particular candidate simply by computing all of them and picking the one that minimises a cost function that measures a certain distance between $\boldsymbol{p}$ and $\boldsymbol{q}$. More precisely, we consider the difference between the variance of the length of their edges. Recall [67] that the variance of the edge lengths of $\boldsymbol{p}$ is defined as

$$\sigma^2(\boldsymbol{p}) = \frac{1}{n+1}\sum_{i=0}^{n}\big(e_i(\boldsymbol{p}) - \overline{e}(\boldsymbol{p})\big)^2,$$

where

$$\overline{e}(\boldsymbol{p}) = \frac{1}{n+1}\sum_{i=0}^{n}e_i(\boldsymbol{p}), \qquad e_i(\boldsymbol{p}) = \|p_{i+1} - p_i\|, \quad i = 0,\dots,n,$$

and similarly for $\boldsymbol{q}$. Among the $n+1$ different degree-elevated control polygons, we then find the one that minimizes

$$|\sigma^2(\boldsymbol{p}) - \sigma^2(\boldsymbol{q})|. \tag{7.12}$$

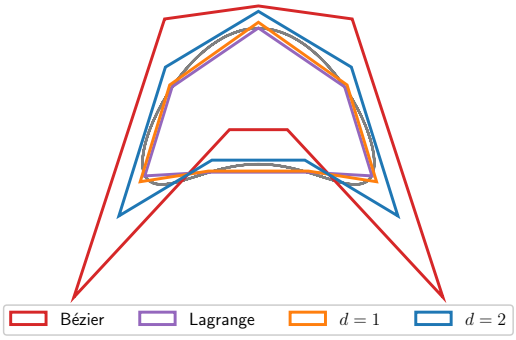


Figure 7.7. Control polygons of the same trigonometric polynomial curve for different sets of basis functions.

Figure 7.6 shows an example of this procedure.

7.2.5 Basis transformations

Each method for designing trigonometric curves (see Figure 7.4) has its particular advantages. Although our type-2 curves might be the preferred representation, since they allow shape control that is very similar to the manipulation of cubic B-spline curves, our type-1 curves have the advantage of tight edge control, the Lagrange basis offers direct vertex control, and the periodic Bézier representation guarantees the convex hull property. Since all representations model the same space of trigonometric polynomials, it is easy to convert between them and to switch from one representation to another, a process also known as basis transformation (see Figure 7.7).

We first study the relation between the Bézier control polygon and the control polygon of our type-1 and type-2 curves. Given a curve $P(t)$ with control points $p_0, \ldots, p_n$ as in (7.3), we would like to find the control points $q_0, \ldots, q_n$, so that the same curve can be expressed in Bézier form as

$$P(t) = \sum_{i=0}^{n} B_i^n(t) q_i.$$

To this end, we express $B_i^n(t)$ in trigonometric polynomial form,

$$B_i^n(t) = \frac{2^n}{n+1} \binom{n}{N}^{-1} \cos^n \frac{t-\phi_i}{2}.$$

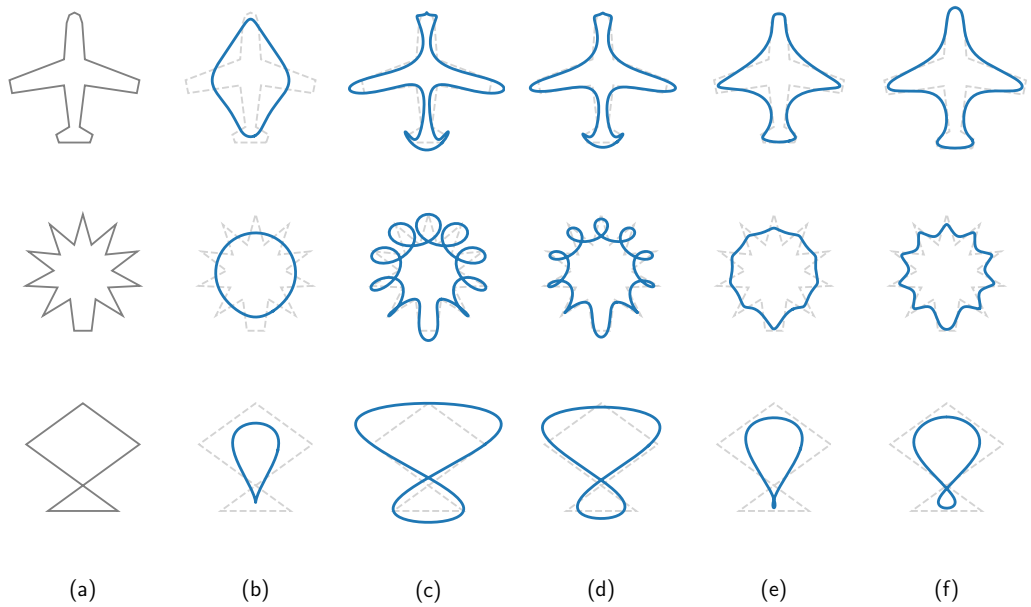


Figure 7.8. Another comparison of periodic Bézier curves (b), vertex interpolating curves (c), our tangent interpolating type-1 (d) and type-2 curves (e), and cubic B-spline curves (f), all defined by the same control polygon (a); cf. Figure 7.1. This example highlights the result given by control polygons with a greatly varying control edge lengths (top), a zig-zag pattern (middle) and a self-intersection (bottom).

We recall the trigonometric power identity [58]

$$\cos^{2m} t = \frac{1}{2^{2m}}\binom{2m}{m} + \frac{1}{2^{2m-1}}\sum_{i=0}^{m-1}\binom{2m}{i}\cos\big(2(m-i)t\big). \tag{7.13}$$

Therefore, we have

$$\begin{aligned} B_i^n(t) &= \frac{1}{n+1}\binom{n}{N}^{-1}\left[\binom{n}{N} + 2\sum_{k=0}^{N-1}\binom{n}{k}\cos[(N-k)(t-\phi_i)]\right] \\ &= \frac{1}{n+1}\binom{n}{N}^{-1}\left[\binom{n}{N} + 2\sum_{k=1}^{N}\binom{n}{N-k}\cos[k(t-\phi_i)]\right] \\ &= \frac{1}{n+1} + \frac{2}{n+1}\binom{n}{N}^{-1}\sum_{k=1}^{N}\binom{n}{N-k}\cos(kt)\cos(k\phi_i) \\ &\qquad + \frac{2}{n+1}\binom{n}{N}^{-1}\sum_{k=1}^{N}\binom{n}{N-k}\sin(kt)\sin(k\phi_i), \end{aligned}$$

and deduce that

$$B_i^n(t) = \frac{1}{n+1}\binom{n}{N}^{-1} \boldsymbol{c}_n(\phi_i)^T D \boldsymbol{c}_n(t),$$

where D is the diagonal matrix

$$D = \operatorname{diag}\left(\binom{n}{N}, 2\binom{n}{N-1}, 2\binom{n}{N-1}, \ldots, 2, 2\right).$$

Letting $\boldsymbol{B}_n(t)$ denote the vector $\boldsymbol{B}_n(t) = \left(B_0^n(t), \ldots, B_n^n(t)\right)^T$, we have $\boldsymbol{B}_n(t) = \Lambda_n \boldsymbol{c}_n(t)$ where

$$\Lambda_n = \frac{1}{n+1}\binom{n}{N}^{-1} C_{n,n} D,$$

which implies

$$\Gamma_{n,d}^{-1} \boldsymbol{L}_n(t) = \Lambda_n^{-1} \boldsymbol{B}_n(t).$$

Consequently, denoting by $\boldsymbol{p}$ and $\boldsymbol{q}$ the vectors $(p_0, \ldots, p_n)$ and $(q_0, \ldots, q_n)$, respectively, we have

$$\boldsymbol{p}\Gamma_{n,d} = \boldsymbol{q}\Lambda_n.$$

Hence, we can switch between the control points of a tangent interpolating curve and the control points of a periodic Bézier curve by solving either for $\boldsymbol{p}$ or for $\boldsymbol{q}$, or rather by multiplying $\boldsymbol{p}$ or $\boldsymbol{q}$ with a precomputed matrix. Similarly, we can switch between the control polygons $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ of the type-1 and type-2 curves, respectively, by solving

$$\boldsymbol{p}_1\Gamma_{n,1} = \boldsymbol{p}_2\Gamma_{n,2}.$$

The conversion between the periodic Bézier and the interpolating Lagrange form is discussed in [94].

Chapter 8

Trigonometric curves given by even number of points

We note that trigonometric curves require $2N+1$ points. However, designing a curve with inherent symmetry can be more intuitive with an even number of control points (see Figure 8.1). The basis functions $B_i^n(t)$ in (2.50) form a basis of the space of trigonometric polynomials of order N. It is preferable that the new space generated by the new functions is a hyperspace of dimension $2N$. We follow the fashion of considering the latter to be the space of balanced trigonometric polynomials as in the literature [10, 63]. We recall that a balanced trigonometric polynomial of order N is of the form

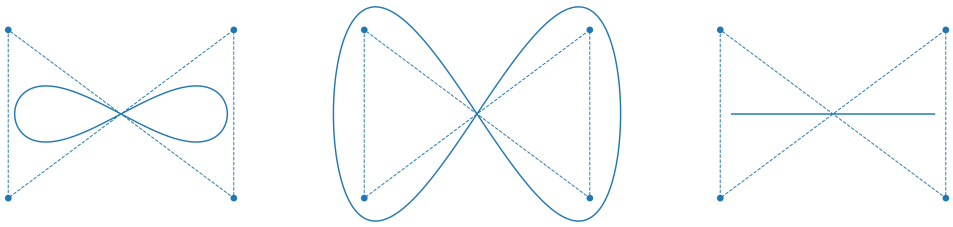
$$p(t) = a_0 + \sum_{k=1}^{N} [a_k \cos kt + b_k \sin kt], \qquad \text{where } b_N = \mathbf{0}, \tag{8.1}$$

for some $a_0, \ldots, a_N, b_1, \ldots, b_{N-1} \in \mathbb{R}^2$. There are two directions that we can take to tackle this problem. The one considers the same basis as in (2.50), (3.5), and (7.11), and translates the condition $b_N = 0$ into a relation between the $2N+1$ control points. The other is to construct a basis for the space of balanced trigonometric polynomials $\operatorname{span}\{1, \cos t, \sin t, \ldots, \cos Nt\}$. We follow this second direction to construct curves given by an even number of points.

8.1 Periodic Bézier curves

Consider $n = 2N-1$ and $\phi_k = \frac{2k\pi}{n+1}$ for $k = 0, \ldots, n$. We start the construction by defining $B_n(t)$ as

$$B_n(t) = \frac{2^{n+1}}{n+1} \binom{n+1}{N}^{-1} \cos^{n+1} \frac{t}{2}. \tag{8.2}$$



(a) (b) (c)

Figure 8.1. This illustrates the shape resulting using (a) periodic Bézier curve, (b) type-1 curve, (c) type-2 curve from the same control polygon. This illustrates the necessity of having curves given by even number of points, since point-symmetric curve are easier to produce. This also show the case where the type-2 curve construction fails for curves given by even-number of control points.

Akin the case in (2.50), we consider the uniform shifts $B_k^n(t) = B_n(t-\phi_k)$, $k = 0,\dots,n$, as candidates for the basis functions. In fact, we have the following.

Proposition 8.1. *The functions $B_k^n(t)$, $k = 0,\dots,n$, form a basis of the space of balanced trigonometric polynomials.*

Proof. In order to show that these functions span the space of balanced trigonometric polynomials, we first show that they belong to the space. To do this, it suffices to show that the function $\cos^{n+1}\frac{t-\phi_k}{2}$ is a balanced trigonometric polynomial of order N, for $k = 0,\dots,n$. Let $k \in \{0,\dots,n\}$. Applying the identity (7.13), we have

$$\cos^{2N}\frac{t-\phi_k}{2} = \frac{1}{2^{2N}}\binom{2N}{N} + \frac{1}{2^{2N-1}}\sum_{i=0}^{N-1}\binom{2N}{i}\cos[(N-i)(t-\phi_k)].$$

Then by applying the change $j = N-i$, we have

$$\cos^{2N}\frac{t-\phi_k}{2} = \frac{1}{2^{2N}}\binom{2N}{N} + \frac{1}{2^{2N-1}}\sum_{j=1}^{N}\binom{2N}{N-j}\cos[j(t-\phi_k)].$$

By expanding the cosine term, we have

$$\cos^{2N}\frac{t-\phi_k}{2} = \frac{1}{2^{2N}}\binom{2N}{N} + \frac{1}{2^{2N-1}}\sum_{j=1}^{N}\binom{2N}{N-j}[\cos jt\,\cos j\phi_k + \sin jt\,\sin j\phi_k]. \tag{8.3}$$

We observe that the highest order in the summation is $\cos Nt\cos N\phi_k + \sin Nt\sin N\phi_k$. In addition, note that $N\phi_k = k\pi$. Hence, we conclude that the coefficient of $\sin Nt$ vanishes.

Now, it remains to show that these functions are linearly independent. Let $\boldsymbol{b}(t) = (B_0^n(t), \ldots, B_n^n(t)^T$. We show that these functions are linearly independent by showing that $\boldsymbol{b}(t) = \Gamma \boldsymbol{c}_n(t)$ for some invertible matrix Γ, and $\boldsymbol{c}_n(t)$ be

$$\boldsymbol{c}_n(t) = \begin{cases} (1, \cos t, \sin t, \ldots, \cos(Nt), \sin(Nt))^T, & \text{if } n = 2N, \\ (1, \cos t, \sin t, \ldots, \cos(Nt))^T, & \text{if } n = 2N-1. \end{cases}$$

We observe that

$$\cos^{2N} \frac{t-\phi_k}{2} = \frac{1}{2^{2N-1}} \boldsymbol{c}_n(\phi_k)\, D\, \boldsymbol{c}^T(t),$$

where D is the diagonal matrix $D = \operatorname{diag}\left(\frac{1}{2}\binom{n}{N}, \binom{n}{N-1}, \binom{n}{N-1}, \ldots, \binom{n}{1}, \binom{n}{1}, 1\right)$. Subsequently, we take

$$\Gamma = \frac{2}{n+1}\binom{n+1}{N}^{-1} \begin{pmatrix} \boldsymbol{c}_n(\phi_0) & \cdots & \boldsymbol{c}_n(\phi_n) \end{pmatrix}^T D. \tag{8.4}$$

The matrix $\begin{pmatrix} \boldsymbol{c}_n(\phi_0) & \cdots & \boldsymbol{c}_n(\phi_n) \end{pmatrix}^T$ is invertible since $\sum_{i=0}^n \phi_i = n\pi \not\equiv 0 \mod (2\pi)$ [10]. We note that D is a diagonal matrix with nonzero diagonal entries. We deduce that Γ is invertible. □

We now instigate the properties of the basis functions. The properties $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_4$ are inherited from $\cos^{n+1}\frac{t}{2}$. The partition of unity $\mathbf{p}_3$ follows from the fact that

$$\sum_{i=0}^n B_i^n(t) = \frac{2^{n+1}}{n+1}\binom{n+1}{N}^{-1} \sum_{i=0}^n \cos^{n+1}\frac{t-\phi_i}{2}.$$

and that the sum $\sum_{i=0}^n \cos^{n+1}\frac{t-\phi_i}{2}$ is given by

$$\begin{aligned} \sum_{i=0}^n \cos^{2N}\frac{t-\phi_i}{2} &\overset{(7.13)}{=} \frac{n+1}{2^{2N}}\binom{2N}{N} + \frac{1}{2^{2N-1}} \sum_{i=0}^{2N-1}\sum_{j=0}^{N-1} \binom{2N}{j} \cos(2(N-j)\frac{t-\phi_i}{2}) \\ &= \frac{n+1}{2^{n+1}}\binom{n+1}{N} + \frac{1}{2^n}\sum_{j=0}^{N-1}\binom{n+1}{j}\sum_{i=0}^{2N-1}\cos((N-j)t - i(N-j)\frac{\pi}{N}) \\ &\overset{(7.13)}{=} \frac{n+1}{2^{n+1}}\binom{n+1}{N} + \frac{1}{2^n}\sum_{j=0}^{N-1}\binom{n+1}{j}\frac{\sin(N-j)\pi\cos((N-j)t - n(N-j)\frac{\pi}{n})}{\sin(N-j)\frac{\pi}{2N}} \\ &= \frac{n+1}{2^{n+1}}\binom{n+1}{N}. \end{aligned}$$

Together with (2.50), we can now define a periodic Bézier curve defined by $p_0, \ldots, p_n$ as

$$p(t) = \sum_{i=0}^{n} B_i^n(t) p_i, \quad \text{where } B_i^n(t) = \begin{cases} \frac{2^{n+1}}{n+1}\binom{n+1}{N}^{-1} \cos^{n+1} \frac{(n+1)t - 2i\pi}{2(n+1)} & \text{if } n = 2N-1, \\ \frac{2^n}{n+1}\binom{n}{N}^{-1} \cos^{n} \frac{(n+1)t - 2i\pi}{2(n+1)} & \text{if } n = 2N. \end{cases} \tag{8.5}$$

8.1.1 Degree elevation

As in Section 2.5.2 and 6.3, this process can be done by matrix multiplication. Denote $\boldsymbol{p}_n = (p_0, \ldots, p_n)^T$. Let $\varphi_k = \frac{2k\pi}{n+2}$, $k = 0, \ldots, n+1$. The degree elevated control polygon can be obtained by solving for $\boldsymbol{p}_{n+1}$ the equation

$$\Lambda \boldsymbol{p}_{n+1} = \Gamma \boldsymbol{p}_n, \qquad \text{where } \Lambda_{ij} = B_j^{n+1}(\varphi_i), \qquad \Gamma_{ij} = B_j^n(\varphi_i),$$

where $\Lambda \in \mathscr{M}_{n+2\times n+2}$, $\Gamma \in \mathscr{M}_{n+2\times n+1}$. It remains to show that the equation admits a solution. If $n = 2N-1$, that is, $n+1 = 2N$, Ramanantoanina and Hormann [94] show that Λ^{-1} is well defined. If $n = 2N-2$, that is, $n+1 = 2N-1$, we observe from (8.4) that

$$\Lambda^T = \frac{2}{n+2}\binom{n+2}{N}^{-1} MDM^T,$$

where $M = \begin{pmatrix} \boldsymbol{c}_{n+1}(\phi_0) & \cdots & \boldsymbol{c}_{n+1}(\phi_{n+1}) \end{pmatrix}^T$ and D as in (8.4) is a diagonal matrix $D = \operatorname{diag}(\frac{1}{2}\binom{n+1}{N}, \binom{n+1}{N-1}, \binom{n+1}{N-1}, \ldots, n+1, n+1, 1)$. Then we deduce that Λ is invertible from the fact that M and D are invertible.

8.1.2 Degree reduction

The degree elevation solves $\Lambda \boldsymbol{p}_{n+1} = \Gamma \boldsymbol{p}_n$ for $\boldsymbol{p}_{n+1}$. The reverse process, which solves for $\boldsymbol{p}_n$, is called *degree reduction*. Since Γ is not a square matrix, we apply a least-square technique to devise $\boldsymbol{p}_n$ from $\boldsymbol{p}_{n+1}$. We recall that the functions $B_k^n(t)$ are linearly independent, therefore, $\Gamma^T\Gamma$ is invertible. We can determine $\boldsymbol{p}_n$ as

$$\boldsymbol{p}_n = (\Gamma^T\Gamma)^{-1}\Gamma^T\Lambda \boldsymbol{p}_{n+1}.$$

For the degree elevation and degree reduction, we can use the same optimisation process in 7.2.4 to select a suitable new control polygon.

8.2 Barycentric rational interpolation

We first discuss how to write a balanced trigonometric polynomial into a barycentric interpolating form. Consider a trigonometric polynomial $p(t)$ as in (8.1). And consider a sequence of parameters $t_0, \ldots, t_n$, and their respective evaluation $p_k = p(t_k)$ for $n = 2N-1$, Salzer [101] shows that we can write $p(t)$ in the following form

$$p(t) = \sum_{i=0}^{n} \ell_i(t) \cos \frac{t-t_i}{2} p_i + c\ell(t), \tag{8.6}$$

where $\ell(t)$ and $\ell_i(t)$ as in (3.24), and c is a coefficient that makes $p(t)$ a balanced polynomial

$$c = (-1)^N 2^n (a_N \cos \frac{\sigma}{2} + b_N \sin \frac{\sigma}{2}), \qquad \sigma = \sum_{i=0}^{n} t_i. \tag{8.7}$$

We can factor out $\ell(t)$ and write $p(t)$ in first barycentric form as

$$p(t) = \ell(t) \left[\sum_{i=0}^{n} \omega_k \cot \frac{t-t_i}{2} p_i + c \right]. \tag{8.8}$$

We can write $p(t)$ in barycentric form by dividing with the interpolation form of the constant function $1 = \ell(t) \sum_{i=0}^{n} \omega_i \cot \frac{t-t_i}{2}$ and we have

$$p(t) = \frac{\sum_{i=0}^{n} \omega_i \cot \frac{t-t_i}{2} p_i + c}{\sum_{i=0}^{n} \omega_i \cot \frac{t-t_i}{2}}.$$

Now we extend this for a periodic rational Bézier function and show that we can essentially convert back and forth between the Bézier and the barycentric form.

Now, consider a periodic rational Bézier $p(t)$ in the form (2.8). In the similar manner as in Chapter 3, $p(t)$ can be written in barycentric form

$$p(t) = \frac{\sum_{i=0}^{n} (-1)^i \cot \frac{t-t_i}{2} v_i q_i + c_q}{\sum_{i=0}^{n} (-1)^i \cot \frac{t-t_i}{2} v_i q_i + c_v} \tag{8.9}$$

for some vector c_q and scalar c_v. The pairs (q_i, v_i) are the same as in (3.28).

For a Bézier curve, from (8.3), we have

$$\begin{pmatrix} c_q \\ c_v \end{pmatrix} = \binom{2N}{N}^{-1} \sum_{i=0}^{n} (-1)^i \widehat{p}.$$

Denote $\widehat{c} = \binom{c_q}{c_v}$. The restriction $b_N = 0$ in (8.1) translates as

$$\cos\frac{\sigma}{2}\sum_{i=0}^{n}(-1)^i\widehat{q}_i = \widehat{c}\sin\frac{\sigma}{2}. \tag{8.10}$$

The issue arises when we want to displace a point or modify a weight since this relation would no longer hold. The interpolation problem can be relaxed by allowing q_{k-1} and q_{k+1} to move optimally when the user manipulates q_k. However, this goes against the whole point of using an interpolating form to be able to make the drawing as precise as possible.

8.3 Trigonometric tangent interpolating curve

We also explore the extension of the family of curves in Chapter 7. The basis functions are computed similarly except for the formula (7.8), the kernel becomes

$$K(t) = \frac{1}{n+1}\left(1 + 2\sum_{k=1}^{N-1}\cos(kt) + \cos Nt\right).$$

Hence the basis functions are given by

$$L_i^d(t) = \frac{1}{n+1} + \frac{2}{d\pi}\sum_{k=1}^{N-1}\frac{1}{k}\cos\big(k(t-\phi_i)\big)\sin\frac{k\phi_d}{2} + \frac{1}{dN\pi}\cos\big(N(t-\phi_i)\big)\sin\frac{N\phi_d}{2}. \tag{8.11}$$

Consequently, the functions $L_i^1(t)$, $i = 0,\dots,n$ form a basis of the space of balanced trigonometric polynomials. For $d = 2$, $\sin\frac{N\phi_2}{2} = 0$. Hence, the basis functions are not linearly independent, and therefore, they are not ideal for curve manipulation (see Figure 8.1.c).

Chapter 9

Surface shape control technique

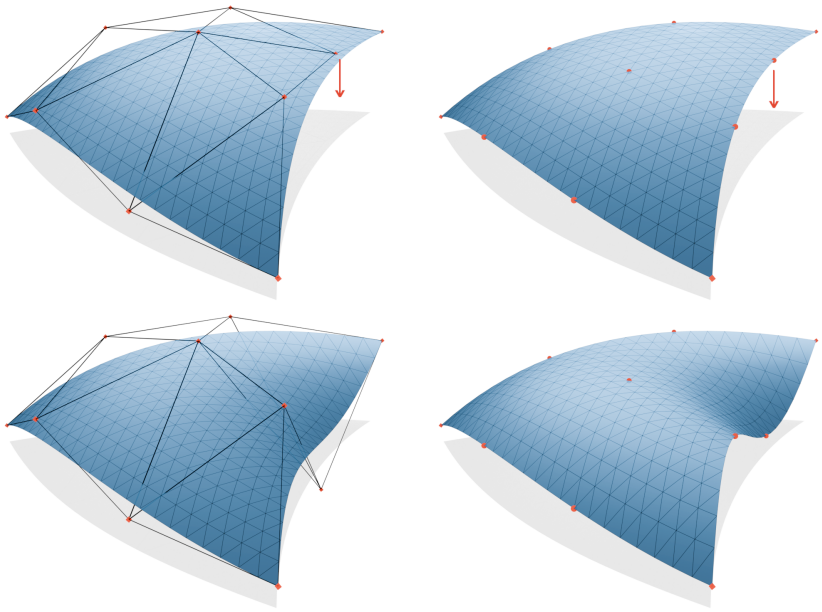


Figure 9.1. A cubic Bézier triangle defined by Bézier points (left) as in (9.1) and by surface points (right) as in (9.3) with respect to the interpolation nodes coinciding with the vertices of a regular grid (top) and effect of modifying one control point (bottom).

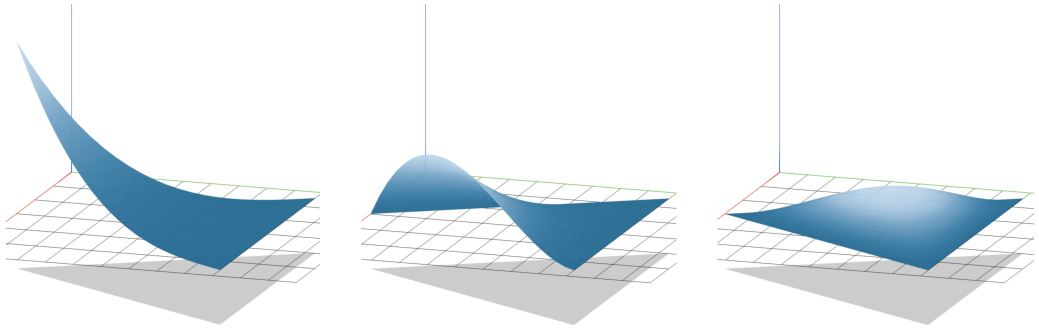


Figure 9.2. Graphs of the cubic Bernstein polynomials $B^3_{003}(\boldsymbol{u}), B^3_{102}(\boldsymbol{u})$, and $B^3_{111}(\boldsymbol{u})$.

9.1 Bézier triangle

A triangular Bézier patch, or simply a *Bézier triangle* [45] of degree n, can be defined in terms of barycentric coordinates $\boldsymbol{u} = (u, v, w)$ with $u + v + w = 1$ as

$$p(\boldsymbol{u}) = \sum_{(i,j,k)\in\mathscr{I}} B^n_{ijk}(\boldsymbol{u}) p_{ijk}, \tag{9.1}$$

for some *Bézier points* $(p_{ijk})_{(i,j,k)\in\mathscr{I}}$ in $\mathbb{R}^3$, where $\mathscr{I}$ is the set of indices

$$\mathscr{I} = \{(i,j,k) : 0 \le i,j,k \le n, i+j+k = n\}$$

and $B^n_{ijk}(\boldsymbol{u})$ is the degree-n Bernstein polynomial (see Figure 9.2)

$$B^n_{ijk}(\boldsymbol{u}) = \frac{n!}{i!j!k!} u^i v^j w^k. \tag{9.2}$$

The Bézier points p_{ijk} form a control net and can be used to intuitively modify the shape of the surface (see Figure 9.1, left), but the control is only *indirect* in the sense that the user does not interact with points that lie on the surface. In analogy to the curve setting [93], it would be nice for the user to also have the possibility of *directly* editing the surface shape (see Figure 9.1, right) by manipulating a set of *surface points* $q_{ijk} = p(\boldsymbol{\lambda}_{ijk})$, characterised by the parameters or *interpolation nodes* $\boldsymbol{\lambda}_{ijk}$ for $(i,j,k) \in \mathscr{I}$.

To better understand the effect of editing surface points, let us represent the Bézier triangle in (9.1) explicitly in interpolating form;

$$p(\boldsymbol{u}) = \sum_{(i,j,k)\in\mathscr{I}} \ell_{ijk}(\boldsymbol{u}) q_{ijk}, \tag{9.3}$$

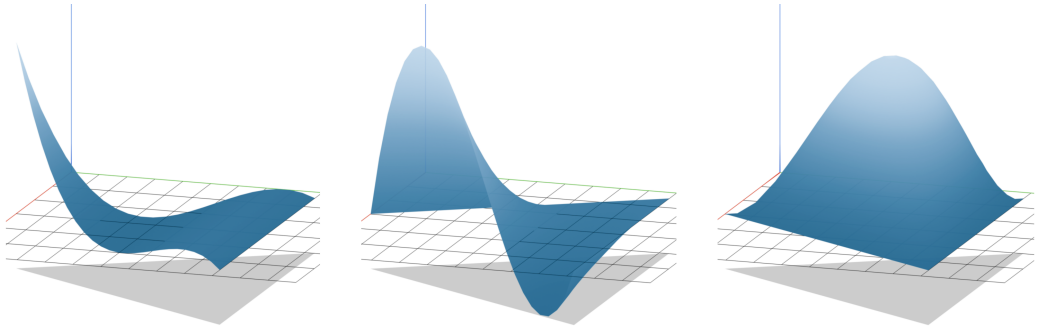


Figure 9.3. Graphs of the cubic Lagrange polynomials $\ell_{003}(\boldsymbol{u}), \ell_{102}(\boldsymbol{u})$, and $\ell_{111}(\boldsymbol{u})$ with respect to a set of interpolation nodes coinciding with the vertices of a regular grid.

where each *Lagrange polynomial* ℓ_{ijk} is a bivariate polynomial of degree n that vanishes at all nodes $\boldsymbol{\lambda}_{abc}$ for $(a,b,c) \in \mathscr{I}$ except at $\boldsymbol{\lambda}_{ijk}$, where it evaluates to 1 (see Figure 9.3). Moving a surface point from its current position q_{ijk} to a new position $\widetilde{q}_{ijk}$ then means that we add the basis function ℓ_{ijk}, scaled by $\Delta^q_{ijk} = \widetilde{q}_{ijk} - q_{ijk}$, to p to get the new surface $\widetilde{p} = p + \ell_{ijk}\Delta^q_{ijk}$. Due to the properties of the Lagrange functions, all other surface points remain in their position during this operation (see Figure 9.1, right).

The concept of Bézier triangles can be extended to the rational setting by introducing a set of weights $(\alpha_{ijk})_{(i,j,k)\in\mathscr{I}}$, each weight associated with a respective control point. A rational Bézier triangle is given by

$$p(\boldsymbol{u}) = \frac{\sum_{(i,j,k)\in\mathscr{I}} B^n_{ijk}(\boldsymbol{u})\alpha_{ijk}p_{ijk}}{\sum_{(i,j,k)\in\mathscr{I}} B^n_{ijk}(\boldsymbol{u})\alpha_{ijk}} \tag{9.4}$$

and can be considered as the central projection under the map

$$\Pi(x,y,z,w) = (x/w, y/w, z/w)$$

of the polynomial Bézier triangle

$$\widehat{p}(\boldsymbol{u}) = \sum_{(i,j,k)\in\mathscr{I}} B^n_{ijk}(\boldsymbol{u})\widehat{p}_{ijk}, \tag{9.5}$$

with homogeneous Bézier points $\widehat{p}_{ijk} = (\alpha_{ijk}p_{ijk}, \alpha_{ijk})$. Increasing or decreasing a weight α_{ijk} has the effect of pulling the surface towards or pushing it away from p_{ijk}.

We can similarly extend the interpolating form (9.3) to the rational setting by associating a weight β_{ijk} with each point q_{ijk}. The rational interpolating function is given by

$$p(\boldsymbol{u}) = \frac{\sum_{(i,j,k)\in\mathscr{I}_n} \ell_{ijk}(\boldsymbol{u})\beta_{ijk}q_{ijk}}{\sum_{(i,j,k)\in\mathscr{I}_n} \ell_{ijk}(\boldsymbol{u})\beta_{ijk}}, \tag{9.6}$$

which can also be considered as the central projection of the polynomial

$$\widehat{p}(\boldsymbol{u}) = \sum_{(i,j,k)\in\mathscr{I}} \ell_{ijk}(\boldsymbol{u})\widehat{q}_{ijk}, \tag{9.7}$$

with homogeneous surface points $\widehat{q}_{ijk} = (\beta_{ijk}q_{ijk}, \beta_{ijk})$. Consequently, we relate the rational forms (9.4) and (9.6) simply by relating their respective homogeneous forms (9.5) and (9.7). We will see that the weights β_{ijk} can be used to control the local flatness of the surface at q_{ijk} (Section 9.5).

9.1.1 Related works

The crucial issue with the interpolation form (9.3) is that Lagrange polynomials ℓ_{ijk} may not be unique or may not even exist for certain configurations of interpolation nodes $\boldsymbol{\lambda}_{ijk}$. In fact, this happens if and only if all $\boldsymbol{\lambda}_{ijk}$ lie on a common algebraic curve of degree n. However, simple geometric conditions have been proposed that guarantee that this is not the case.

The simplest condition of this kind is that the $\boldsymbol{\lambda}_{ijk}$ form a *triangular grid* [57].

Geometric characterisation GC 1. There exist three sets of lines $\{R_0,\dots,R_n\}$, $\{S_0,\dots,S_n\}$, and $\{T_0,\dots,T_n\}$ such that each $\boldsymbol{\lambda}_{ijk}$ is the common intersection point of the lines R_i, S_j, and T_k.

In this situation (see Figure 9.4, left), we can derive an explicit formula for the Lagrange polynomials as products of n linear functions by first renaming the lines R_l, S_l, and T_l for $l = 1,\dots,n$ as $L_{(n-l)00}$, $L_{0(n-l)0}$, and $L_{00(n-l)}$, respectively, and expressing them in barycentric coordinates as

$$\begin{aligned} L_{(n-l)00} &= \boldsymbol{\lambda}_{l(n-l)0} \times \boldsymbol{\lambda}_{l0(n-l)}, \\ L_{0(n-l)0} &= \boldsymbol{\lambda}_{(n-l)l0} \times \boldsymbol{\lambda}_{0l(n-l)}, \\ L_{00(n-l)} &= \boldsymbol{\lambda}_{0(n-l)l} \times \boldsymbol{\lambda}_{(n-l)0l}, \end{aligned}$$

and then normalizing the degree-n polynomials

$$r_{ijk}(\boldsymbol{u}) = \prod_{l=0}^{i-1}(L_{(n-l)00}\cdot\boldsymbol{u})\prod_{l=0}^{j-1}(L_{0(n-l)0}\cdot\boldsymbol{u})\prod_{l=0}^{k-1}(L_{00(n-l)}\cdot\boldsymbol{u}), \tag{9.8}$$

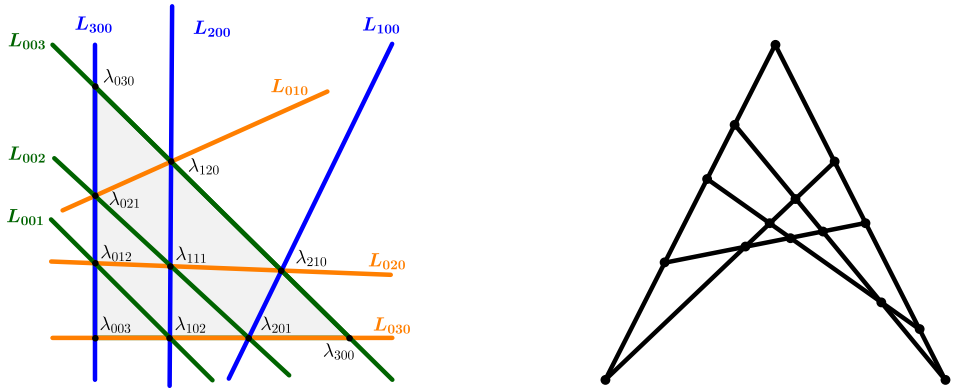


Figure 9.4. Examples of node configurations satisfying GC1 for $n = 3$ (left) and GC2 for $n = 4$ (right).

which are products of exactly n lines to get

$$\ell_{ijk}(\boldsymbol{u}) = \frac{r_{ijk}(\boldsymbol{u})}{r_{ijk}(\boldsymbol{\lambda}_{ijk})}$$

for $(i,j,k) \in \mathscr{I}$. For uniformly distributed nodes, the basis functions can be written explicitly as [21]

$$\ell_{ijk}(\boldsymbol{u}) = \frac{n^{-n}}{i!j!k!} \prod_{l=0}^{i-1}(u-l/n) \prod_{l=0}^{j-1}(v-l/n) \prod_{l=0}^{k-1}(w-l/n), \qquad \boldsymbol{u} = (u,v,w).$$

The previous construction gives Lagrange polynomials, because the n lines involved in the definition of p_{ijk} in (9.8) contain all interpolation nodes except $\boldsymbol{\lambda}_{ijk}$. This observation leads to the following, more general condition by [32] (see Figure 9.4, right).

Geometric characterisation GC 2. For each $\boldsymbol{\lambda}_{ijk}$, there exist n lines that contain all nodes except $\boldsymbol{\lambda}_{ijk}$.

For even more general node configurations, we can express $\boldsymbol{b}$ in interpolating form by writing the Lagrange polynomials in terms of Vandermonde matrices. We define a generalized Vandermonde matrix for a set of parameter points $\{\boldsymbol{u}_{ijk} : (i,j,k) \in \mathscr{I}\}$ as[1]

$$V(\boldsymbol{u}_{00n}, \ldots, \boldsymbol{u}_{n00}) = \left(B^n_{ijk}(\boldsymbol{u}_{abc})\right)_{(i,j,k),(a,b,c)\in\mathscr{I}}. \tag{9.9}$$

[1] To enumerate over all elements of $\mathscr{I}$, we use lexicographical order. For example, if $n = 2$, then we have $(\boldsymbol{u}_{002}, \ldots, \boldsymbol{u}_{200}) = (\boldsymbol{u}_{002}, \boldsymbol{u}_{011}, \boldsymbol{u}_{020}, \boldsymbol{u}_{101}, \boldsymbol{u}_{110}, \boldsymbol{u}_{200})$.

Letting

$$\boldsymbol{p} = \begin{pmatrix} p_{00n} \\ \vdots \\ p_{n00} \end{pmatrix}, \qquad \boldsymbol{q} = \begin{pmatrix} q_{00n} \\ \vdots \\ q_{n00} \end{pmatrix},$$

we can write the interpolation conditions $p(\boldsymbol{\lambda}_{ijk}) = q_{ijk}$ for $(i,j,k) \in \mathscr{I}$ compactly as

$$V(\boldsymbol{\lambda}_{00n}, \ldots, \boldsymbol{\lambda}_{n00})\boldsymbol{p} = \boldsymbol{q}. \tag{9.10}$$

Hence, as long as $\det V(\boldsymbol{\lambda}_{00n}, \ldots, \boldsymbol{\lambda}_{n00}) \neq 0$ [107], that is, the nodes does not belong on an algebraic hypersurface of degree n [81], the Lagrange polynomials can be written as

$$\ell_{ijk}(\boldsymbol{u}) = \frac{\det V(\boldsymbol{\lambda}_{00n}, \ldots, \boldsymbol{u}, \ldots, \boldsymbol{\lambda}_{n00})}{\det V(\boldsymbol{\lambda}_{00n}, \ldots, \boldsymbol{\lambda}_{ijk}, \ldots, \boldsymbol{\lambda}_{n00})}. \tag{9.11}$$

There exists several particular sets of nodes where this is the case and used for polynomial approximation [19, 24, 30, 65, 117, 122]. Furthermore, we can use this observation to convert the surface from the interpolating form (9.3) into Bézier form (9.1) using (9.10), but this naive approach requires solving a linear system of size $(n+1)(n+2)/2$.

9.2 Quadratic interpolation

For quadratic surfaces ($n = 2$), all Bézier points belong to the boundary of the control net. Hence, it is natural to also place the surface points along the boundary of the patch, mimicking the positions of the Bézier points and providing full control over the shapes of the three boundary curves. In particular, we propose having three corner points at the fixed interpolation nodes $\boldsymbol{\lambda}_{002} = (0,0,1)$, $\boldsymbol{\lambda}_{020} = (0,1,0)$, and $\boldsymbol{\lambda}_{200} = (1,0,0)$ and three edge points at $\boldsymbol{\lambda}_{011} = (0, 1-\rho, \rho)$, $\boldsymbol{\lambda}_{101} = (\sigma, 0, 1-\sigma)$, and $\boldsymbol{\lambda}_{110} = (1-\tau, \tau, 0)$ for some $\rho, \sigma, \tau \in (0,1)$. This means that the interpolation nodes are given by a triangular grid (see Figure 9.5) and satisfy GC1. Hence, the Lagrange polynomials are well defined, and the interpolation problem (9.10) is unisolvent. During modelling, the user can *slide* the edge surface points q_{011}, q_{101}, and q_{110} along the respective boundary curve and freely move each surface point.

For this node configuration, it is possible to update the Bézier points of the surface, after a modification of a surface point, more efficiently than by solving the linear system in (9.10). To this end, we denote by "$\widetilde{\cdot}$" the modified or

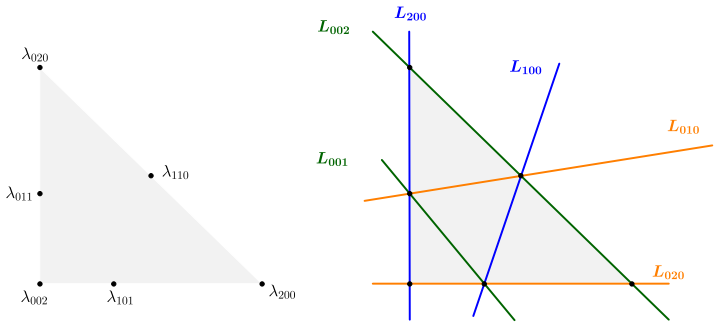


Figure 9.5. The proposed configuration of nodes for the quadratic case (left) and the respective set of lines from GC1 (right).

updated point with respect to a point " $\cdot$ " which may refer to either a Bézier control point or a surface point.

First, assume that we modify a corner point. Without loss of generality, we change q_{002} to $\widetilde{q}_{002}$. As B^2_{002} is the only quadratic Bernstein polynomial that is nonzero at $\boldsymbol{\lambda}_{002}$ with $B^2_{002}(\boldsymbol{\lambda}_{002}) = 1$, we have to set $\widetilde{p}_{002} = \widetilde{q}_{002}$. Moreover, we know that $\ell_{002}(\boldsymbol{u}) = 0$ for $\boldsymbol{u} = (u, 1-u, 0)$. This means that the opposite edge boundary curve, that is, the edge supported by L_{002} in Figure 9.5, is not affected by the manipulation of q_{002}. Hence, it remains to determine $\widetilde{p}_{011}$ and $\widetilde{p}_{101}$. Let $\Delta^p_{ijk} = \widetilde{p}_{ijk} - p_{ijk}$ and $\Delta^q_{ijk} = \widetilde{q}_{ijk} - q_{ijk}$. By expanding the relation $\widetilde{p}(\boldsymbol{\lambda}_{101}) - p(\boldsymbol{\lambda}_{101}) = \Delta^q_{101} = 0$, we deduce that

$$\Delta^p_{101} = -\frac{B^2_{002}(\boldsymbol{\lambda}_{101})}{B^2_{101}(\boldsymbol{\lambda}_{101})}\Delta^p_{002} \tag{9.12}$$

and similarly for Δ^p_{011}.

Second, if we update, without loss of generality, the edge point q_{101} to $\widetilde{q}_{101}$, then $\widetilde{p}_{101} = p_{101} + \Delta^p_{101}$ with

$$\Delta^p_{101} = \frac{1}{B^2_{101}(\boldsymbol{\lambda}_{101})}\Delta^q_{101}, \tag{9.13}$$

and all other Bézier points remain unchanged, because both ℓ_{101} and B^2_{101} vanish on the edges supported by the lines L_{002} and L_{200}.

9.3 Cubic interpolation

For cubic surfaces ($n = 3$), we could also stick to configurations that satisfy GC1, but as shown in Figure 9.4 (left), this means that L_{002}, L_{020}, and L_{200} must always

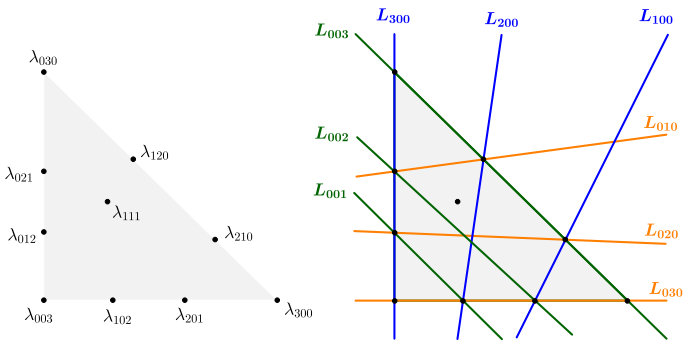


Figure 9.6. An example of the configuration GC3.

intersect at $\boldsymbol{\lambda}_{111}$, so we cannot slide the central surface point q_{111} to an arbitrary position before modifying it. This limitation can be overcome by considering the following geometric condition, which is more general than GC1 (see Figure 9.6).

Geometric characterisation GC 3.

1. Fixed corner nodes $\boldsymbol{\lambda}_{003} = (0,0,1)$, $\boldsymbol{\lambda}_{030} = (0,1,0)$, and $\boldsymbol{\lambda}_{300} = (1,0,0)$.

2. The edge nodes $\boldsymbol{\lambda}_{ij0}, \boldsymbol{\lambda}_{i0k}$, and $\boldsymbol{\lambda}_{0jk}$ are distinct and, respectively, strict convex combinations of $\boldsymbol{\lambda}_{300}$ and $\boldsymbol{\lambda}_{030}$, of $\boldsymbol{\lambda}_{300}$ and $\boldsymbol{\lambda}_{003}$, and of $\boldsymbol{\lambda}_{030}$ and $\boldsymbol{\lambda}_{003}$.

3. The central node is a strict convex combination of the corner nodes $\boldsymbol{\lambda}_{003}$, $\boldsymbol{\lambda}_{030}$, and $\boldsymbol{\lambda}_{300}$, that is, $\boldsymbol{\lambda}_{111} = u\boldsymbol{\lambda}_{003} + v\boldsymbol{\lambda}_{300} + w\boldsymbol{\lambda}_{030}$ for some $u, v, w > 0$ with $u + v + w = 1$.

Note that GC3 is also more general than GC2 but still guarantees the existence of well-defined Lagrange polynomials.

Proposition 9.1. *The interpolation problem* (9.10) *is unisolvent if the interpolation nodes satisfy GC3.*

Proof. If the Vandermonde matrix $V(\boldsymbol{\lambda}_{003}, \ldots, \boldsymbol{\lambda}_{300})$ is singular, this means that there exists a cubic algebraic curve $\mathscr{C}$ that passes through all nodes. By Bézout's theorem [54], since the line L_{300} and the curve $\mathscr{C}$ intersect at $n+1$ nodes, then $L_{003} \subseteq \mathscr{C}$. Similarly, the lines L_{030} and L_{300} are contained in $\mathscr{C}$. Since $\mathscr{C}$ is a cubic curve, this means that $\mathscr{C} = L_{003} \cup L_{030} \cup L_{300}$. This contradicts the fact that $\boldsymbol{\lambda}_{111} \in \mathscr{C}$. Hence, $\det V(\boldsymbol{\lambda}_{003}, \ldots, \boldsymbol{\lambda}_{300}) \neq 0$. □

Furthermore, we can express the Lagrange functions ℓ_{ijk} in (9.11) as

$$\begin{aligned}\ell_{111}(\boldsymbol{u}) &= \frac{r_{111}(\boldsymbol{u})}{r_{111}(\boldsymbol{\lambda}_{111})},\\ \ell_{ijk}(\boldsymbol{u}) &= \frac{r_{ijk}(\boldsymbol{u})}{r_{ijk}(\boldsymbol{\lambda}_{ijk})} - \frac{r_{ijk}(\boldsymbol{\lambda}_{111})}{r_{ijk}(\boldsymbol{\lambda}_{ijk})}\ell_{111}(\boldsymbol{u}), \quad (i,j,k) \neq (1,1,1).\end{aligned} \tag{9.14}$$

Now we can discuss the correspondence between surface and Bézier points and simplify the relation (9.10) accordingly. We start by analysing the changes when we modify a corner point. Assume that, without loss of generality, we change q_{003} to $\widetilde{q}_{003}$. As in the quadratic case, it is clear that $\widetilde{p}_{003} = \widetilde{q}_{003}$. The other Bézier points affected by this manipulation are those located on the two edge boundary curves that contain q_{003} and the central point, that is, the points with indices in

$$\mathscr{I}' = \{(1,0,2),(2,0,1),(0,1,2),(0,2,1),(1,1,1)\}.$$

Letting $\Delta^p = (\Delta^p_{abc})_{(a,b,c)\in\mathscr{I}'}$ and $\Delta^q = (\Delta^q_{abc} - B^3_{003}(\boldsymbol{\lambda}_{abc})\Delta^p_{003})_{(a,b,c)\in\mathscr{I}'}$, it follows that (9.10) simplifies to $V'\Delta^p = \Delta^q$ with

$$V' = \begin{pmatrix} A & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B & \mathbf{0} \\ B^3_{102}(\boldsymbol{\lambda}_{111}) \;\; B^3_{201}(\boldsymbol{\lambda}_{111}) & B^3_{012}(\boldsymbol{\lambda}_{111}) \;\; B^3_{021}(\boldsymbol{\lambda}_{111}) & B^3_{111}(\boldsymbol{\lambda}_{111}) \end{pmatrix} \tag{9.15}$$

and

$$A = \begin{pmatrix} B^3_{102}(\boldsymbol{\lambda}_{102}) & B^3_{201}(\boldsymbol{\lambda}_{102}) \\ B^3_{102}(\boldsymbol{\lambda}_{201}) & B^3_{201}(\boldsymbol{\lambda}_{201}) \end{pmatrix}, \qquad B = \begin{pmatrix} B^3_{012}(\boldsymbol{\lambda}_{012}) & B^3_{021}(\boldsymbol{\lambda}_{012}) \\ B^3_{012}(\boldsymbol{\lambda}_{021}) & B^3_{021}(\boldsymbol{\lambda}_{021}) \end{pmatrix}.$$

Proposition 9.2. *The matrix V' in* (9.15) *is invertible.*

Proof. It suffices to show that A and B are invertible. Let $\mathscr{I}''$ and V'' be such that

$$\begin{aligned}\mathscr{I}'' &= \{(0,0,3),(1,0,2),(2,0,1),(3,0,0)\},\\ V'' &= \left(B^3_{ijk}(\boldsymbol{\lambda}_{abc})\right)_{(i,j,k),(a,b,c)\in\mathscr{I}''}.\end{aligned}$$

Since the interpolation nodes with indices in $\mathscr{I}''$ are collinear, V'' can be considered a univariate Vandermonde matrix. This implies that V'' is a totally positive matrix [55]. We note that A is a minor of V'' by excluding the first and last rows, as well as the first and last columns. Therefore, A is invertible. The matrix B can be proven to be invertible in a similar way. □

Since A and B are invertible, we can further decompose the linear system $V'\Delta^p = \Delta^q$ by first extracting the displacement vectors of the Bézier points of the edge boundary curves supported by L_{030} and L_{300} independently and then determining the displacement vector of the central point,

$$\begin{aligned}
\begin{pmatrix} \Delta^p_{102} \\ \Delta^p_{201} \end{pmatrix} &= -A^{-1} \begin{pmatrix} B^3_{003}(\boldsymbol{\lambda}_{102}) \\ B^3_{003}(\boldsymbol{\lambda}_{201}) \end{pmatrix} \Delta^p_{003}, \\
\begin{pmatrix} \Delta^p_{012} \\ \Delta^p_{021} \end{pmatrix} &= -B^{-1} \begin{pmatrix} B^3_{003}(\boldsymbol{\lambda}_{012}) \\ B^3_{003}(\boldsymbol{\lambda}_{021}) \end{pmatrix} \Delta^p_{003}, \\
\Delta^p_{111} &= -\frac{1}{B^3_{111}(\boldsymbol{\lambda}_{111})} \sum_{(i,j,k)\in\mathscr{I}'\setminus\{(1,1,1)\}} B^3_{ijk}(\boldsymbol{\lambda}_{111})\Delta^p_{ijk}.
\end{aligned} \tag{9.16}$$

Now, without loss of generality, assume that we change the edge point q_{102} to $\widetilde{q}_{102}$. Then the only points affected by this manipulation are the Bézier points of the edge boundary curve supported by L_{030} and the central point. This means that we arrive at a similar linear system as before, but with

$$\begin{aligned}
\mathscr{I}' &= \{(1,0,2),(2,0,1),(1,1,1)\}, \\
V' &= \begin{pmatrix} & A & & \mathbf{0} \\ B^3_{102}(\boldsymbol{\lambda}_{111}) & & B^3_{201}(\boldsymbol{\lambda}_{111}) & B^3_{111}(\boldsymbol{\lambda}_{111}) \end{pmatrix}.
\end{aligned}$$

Again, this linear system can be decomposed and solved as in (9.16). The other cases can be treated similarly.

Finally, if we change the central point q_{111} to $\widetilde{q}_{111}$, then we have

$$\Delta^p_{111} = \frac{1}{B^3_{111}(\boldsymbol{\lambda}_{111})}\Delta^q_{111},$$

and all other Bézier points remain unchanged, because both ℓ_{111} and B^3_{111} vanish on the edges supported by the lines L_{003}, L_{030}, and L_{300}.

9.4 Quartic interpolation

For quartic surfaces ($n = 4$), we can still describe a simple configuration of the nodes so that the interpolation problem in (9.10) is solvable, analogue to the geometric condition GC3.

Geometric characterisation GC 4.

1. Fixed corner nodes $\boldsymbol{\lambda}_{004} = (0,0,1)$, $\boldsymbol{\lambda}_{040} = (0,1,0)$, and $\boldsymbol{\lambda}_{400} = (1,0,0)$.

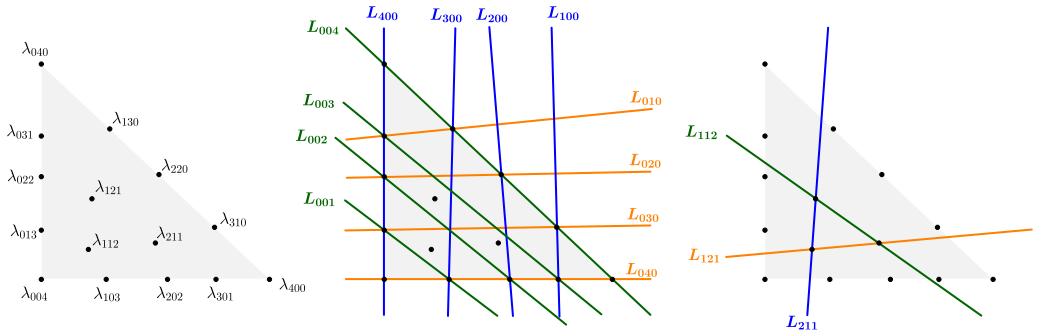


Figure 9.7. An example of the configuration GC4.

2. The edge nodes $\boldsymbol{\lambda}_{ij0}, \boldsymbol{\lambda}_{i0k}$, and $\boldsymbol{\lambda}_{0jk}$ are distinct and respectively strict convex combinations of $\boldsymbol{\lambda}_{400}$ and $\boldsymbol{\lambda}_{040}$, of $\boldsymbol{\lambda}_{400}$ and $\boldsymbol{\lambda}_{004}$, and of $\boldsymbol{\lambda}_{040}$ and $\boldsymbol{\lambda}_{004}$.

3. The three central nodes $\boldsymbol{\lambda}_{112}$, $\boldsymbol{\lambda}_{121}$, and $\boldsymbol{\lambda}_{211}$ are strict convex combinations of $\boldsymbol{\lambda}_{004}$, $\boldsymbol{\lambda}_{040}$, and $\boldsymbol{\lambda}_{400}$ and not collinear.

Proposition 9.3. *The interpolation problem* (9.10) *is unisolvent if the interpolation nodes satisfy GC4.*

Proof. If the Vandermonde matrix $V(\boldsymbol{\lambda}_{004}, \dots, \boldsymbol{\lambda}_{400})$ is singular, then this implies the existence of a quartic algebraic curve $\mathscr{C}$ that passes through all nodes. According to Bézout's theorem [54], this means that the lines L_{004}, L_{040}, and L_{400} are contained in $\mathscr{C}$. However, this scenario is impossible because it means that $\mathscr{C} \setminus (L_{004} \cup L_{040} \cup L_{400})$ is empty or supported by a line. This contradicts the structure of our configuration, as the nodes $\boldsymbol{\lambda}_{112}$, $\boldsymbol{\lambda}_{121}$, and $\boldsymbol{\lambda}_{211}$ do not belong to any of these three lines and are not collinear. Hence, such a quartic curve $\mathscr{C}$ does not exist. □

Now, let us define the index sets

$$\begin{aligned}\mathscr{I}_1 &= \{(1,0,3),(2,0,2),(3,0,1)\},\\ \mathscr{I}_2 &= \{(0,1,3),(0,2,2),(0,3,1)\},\\ \mathscr{I}_3 &= \{(1,1,2),(1,2,1),(2,1,1)\}.\end{aligned}$$

and let $\mathscr{I}' = \mathscr{I}_1 \cup \mathscr{I}_2 \cup \mathscr{I}_3$. By defining p_{ijk} for $(i,j,k) \in \mathscr{I}_3$ as

$$r_{ijk}(\boldsymbol{u}) = (L_{00n} \cdot \boldsymbol{u})(L_{n00} \cdot \boldsymbol{u})(L_{0n0} \cdot \boldsymbol{u})(L_{ijk} \cdot \boldsymbol{u}), \tag{9.17}$$

we can write the Lagrange functions (9.11) as

$$\begin{aligned}\ell_{ijk}(\boldsymbol{u}) &= \frac{r_{ijk}(\boldsymbol{u})}{r_{ijk}(\boldsymbol{\lambda}_{ijk})}, \quad (i,j,k) \in \mathscr{I}_3,\\ \ell_{ijk}(\boldsymbol{u}) &= \frac{r_{ijk}(\boldsymbol{u})}{r_{ijk}(\boldsymbol{\lambda}_{ijk})} - \sum_{(a,b,c)\in\mathscr{I}_3} \frac{r_{ijk}(\boldsymbol{\lambda}_{abc})}{r_{ijk}(\boldsymbol{\lambda}_{ijk})}\ell_{abc}(\boldsymbol{u}), \quad (i,j,k) \in \mathscr{I} \setminus \mathscr{I}_3.\end{aligned} \tag{9.18}$$

Let $\Delta^p = (\Delta^p_{abc})_{(a,b,c)\in\mathscr{I}'}$, and $\Delta^q = (\Delta^q_{abc} - B^4_{004}(\boldsymbol{\lambda}_{abc})\Delta^p_{004})_{(a,b,c)\in\mathscr{I}'}$. As in the previous cases, if we manipulate q_{004}, since the corner Bézier points are interpolated, we have $\widetilde{p}_{004} = \widetilde{q}_{004}$. It remains to determine the new positions of the other affected Bézier points, which are the control points of the edge boundary curves supported by L_{040} and L_{400} and the central points. We can determine the new Bézier points from the relation $V'\Delta^p = \Delta^q$ with

$$\begin{aligned}V' &= \begin{pmatrix} A & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B & \mathbf{0} \\ C & D & E \end{pmatrix},\\ A &= \left(B^4_{ijk}(\boldsymbol{\lambda}_{abc})\right)_{(i,j,k)\in\mathscr{I}_1,(a,b,c)\in\mathscr{I}_2}, \qquad B = \left(B^4_{ijk}(\boldsymbol{\lambda}_{abc})\right)_{(i,j,k)\in\mathscr{I}_2,(a,b,c)\in\mathscr{I}_1},\\ C &= \left(B^4_{ijk}(\boldsymbol{\lambda}_{abc})\right)_{(i,j,k)\in\mathscr{I}_1,(a,b,c)\in\mathscr{I}_3}, \qquad D = \left(B^4_{ijk}(\boldsymbol{\lambda}_{abc})\right)_{(i,j,k)\in\mathscr{I}_2,(a,b,c)\in\mathscr{I}_3},\\ E &= \left(B^4_{ijk}(\boldsymbol{\lambda}_{abc})\right)_{(i,j,k)\in\mathscr{I}_3,(a,b,c)\in\mathscr{I}_3}.\end{aligned} \tag{9.19}$$

Here, it can be shown that V' is also invertible in the same way as in Proposition 9.2 and that we can solve the problem progressively as in (9.16).

If we change the edge point q_{103} to $\widetilde{q}_{103}$, we arrive at a similar linear system as before, but with $\mathscr{I}' = \mathscr{I}_1 \cup \mathscr{I}_3$, $\Delta^q = (\Delta^q_{abc})_{(a,b,c)\in\mathscr{I}'}$, and

$$V' = \begin{pmatrix} A & \mathbf{0} \\ C & E. \end{pmatrix}$$

If we update a central point, we arrive again at a linear system similar to before, but with $V' = E$ and $\mathscr{I}' = \mathscr{I}_3$. The other cases can be obtained similarly.

9.4.1 Degenerate configurations

In practice, we would like to avoid the collinearity constraint on the central nodes in GC4 and allow the user to freely slide the surface points q_{112}, q_{121}, and q_{211} to arbitrary positions, as long as they remain distinct. Hence, it can happen that the interpolation nodes $\boldsymbol{\lambda}_{112}$, $\boldsymbol{\lambda}_{211}$, and $\boldsymbol{\lambda}_{121}$ become collinear.

Geometric characterisation GC 5.

1. Fixed corner nodes $\boldsymbol{\lambda}_{004} = (0,0,1)$, $\boldsymbol{\lambda}_{040} = (0,1,0)$, and $\boldsymbol{\lambda}_{400} = (1,0,0)$.

2. The edge nodes $\boldsymbol{\lambda}_{ij0}, \boldsymbol{\lambda}_{i0k}$, and $\boldsymbol{\lambda}_{0jk}$ are distinct and, respectively, strict convex combinations of $\boldsymbol{\lambda}_{400}$ and $\boldsymbol{\lambda}_{040}$, of $\boldsymbol{\lambda}_{400}$ and $\boldsymbol{\lambda}_{004}$, and of $\boldsymbol{\lambda}_{040}$ and $\boldsymbol{\lambda}_{004}$.

3. The three central nodes $\boldsymbol{\lambda}_{112}$, $\boldsymbol{\lambda}_{121}$, and $\boldsymbol{\lambda}_{211}$ are strict convex combinations of $\boldsymbol{\lambda}_{004}$, $\boldsymbol{\lambda}_{040}$, and $\boldsymbol{\lambda}_{400}$ and collinear.

We know that when the nodes satisfy GC5, there can be either no solution to the interpolation problem or there are infinitely many possibilities. In this case, if we manipulate a corner point or an edge point, we propose to use the Moore–Penrose inverse [69] $V^\dagger$ of V' to extract Δ^p from the relation $V'\Delta^p = \Delta^q$. We note that Δ^p contains the displacement vectors of the Bézier points. This justifies the use of the Moore–Penrose inverse, since $\Delta^p = V^\dagger \Delta^q$ is the best approximate solution that minimises the perturbation of the interpolation points, in case the problem is inconsistent, while ensuring a minimal displacement of the Bézier points [88].

If we modify a central point, we want the modified point to remain interpolated. Hence, simply taking the pseudo-inverse does not suffice anymore. Assume that, without loss of generality, we change q_{112} to $\widetilde{q}_{112}$. Let V' be a Vandermonde matrix and v_1, v_2, v_3 be row vectors such that

$$V' = \begin{pmatrix} v_1^T \\ v_2^T \\ v_3^T \end{pmatrix} = \left(B^4_{ijk}(\boldsymbol{\lambda}_{abc})\right)_{(i,j,k)\in\mathscr{I}_3,(a,b,c)\in\mathscr{I}_3}. \tag{9.20}$$

We suggest altering the interpolation problem by efficiently finding Δ^q_{211} and Δ^q_{121}, thus minimising $\|\Delta^q_{211}\|^2 + \|\Delta^q_{121}\|^2$, to ensure that the system $V'\Delta^p = \Delta^q$ admits an exact solution.

Proposition 9.4. *Assume that we change q_{112} to $\widetilde{q}_{112}$. The optimal solution Δ^p induced by the above constrained least-squares problem is given by*

$$\Delta^p = \frac{1}{\mu_1^2 + \mu_2^2} \begin{pmatrix} v_2^T \\ v_3^T \end{pmatrix}^\dagger \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \Delta^q_{112},$$

for some $\mu_1, \mu_2 \in \mathbb{R}$ such that $v_1 = \mu_1 v_2 + \mu_2 v_3$.

Proof. We find x_1 and x_2 minimising $\|x_1\|^2 + \|x_2\|^2$ such that

$$V'\Delta^p = \begin{pmatrix} \Delta^q_{112} \\ x_1 \\ x_2 \end{pmatrix}.$$

Since V' is a singular matrix, there exist μ_1 and μ_2 such that $v_1 = \mu_1 v_2 + \mu_2 v_3$. Therefore, we can explicitly write the optimisation constraint as $\mu_1 x_1 + \mu_2 x_2 = \Delta^q_{112}$. By considering the Lagrangian function [23]

$$F(x_1, x_2, \zeta) = x_1^T x_1 + x_2^T x_2 + \zeta(\mu_1 x_1 + \mu_2 x_2 - \Delta^q_{112}),$$

where ζ is a row vector, we conclude that

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \frac{1}{\mu_1^2 + \mu_2^2} \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \Delta^q_{112}.$$

Now, since we target $\begin{pmatrix} v_2^T \\ v_3^T \end{pmatrix} \Delta^p = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, the proposition follows by solving for Δ^p in this equation. □

Note that we can decompose V' as $V' = DA$ where

$$A = \begin{pmatrix} B^1_{001}(\boldsymbol{\lambda}_{112}) & B^1_{100}(\boldsymbol{\lambda}_{112}) & B^1_{010}(\boldsymbol{\lambda}_{112}) \\ B^1_{001}(\boldsymbol{\lambda}_{211}) & B^1_{100}(\boldsymbol{\lambda}_{211}) & B^1_{010}(\boldsymbol{\lambda}_{211}) \\ B^1_{001}(\boldsymbol{\lambda}_{121}) & B^1_{100}(\boldsymbol{\lambda}_{121}) & B^1_{010}(\boldsymbol{\lambda}_{121}) \end{pmatrix}, \quad D = 2 \begin{pmatrix} B^3_{111}(\boldsymbol{\lambda}_{112}) & 0 & 0 \\ 0 & B^3_{111}(\boldsymbol{\lambda}_{211}) & 0 \\ 0 & 0 & B^3_{111}(\boldsymbol{\lambda}_{121}) \end{pmatrix}.$$

Therefore, we have

$$\mu_1 = (1-s)\frac{B^3_{111}(\boldsymbol{\lambda}_{112})}{B^3_{111}(\boldsymbol{\lambda}_{211})}, \qquad \mu_2 = s\frac{B^3_{111}(\boldsymbol{\lambda}_{112})}{B^3_{111}(\boldsymbol{\lambda}_{211})},$$

for some s such that $\boldsymbol{\lambda}_{112} = \boldsymbol{\lambda}_{211}(1-s) + s\boldsymbol{\lambda}_{121}$.

Proposition 9.4 leads to a novel approach for handling central points. As the determinant of the Vandermonde matrix (9.20) becomes close but not equal to zero, the manipulation effect becomes increasingly counter-intuitive. Hence, we propose the following adjustments. First, we consider the solution considering that the interpolation problem is not perturbed. Let Δ^p_1 be the exact solution

$$\Delta^p_1 = E^{-1} \begin{pmatrix} \Delta^q_{112} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix},$$

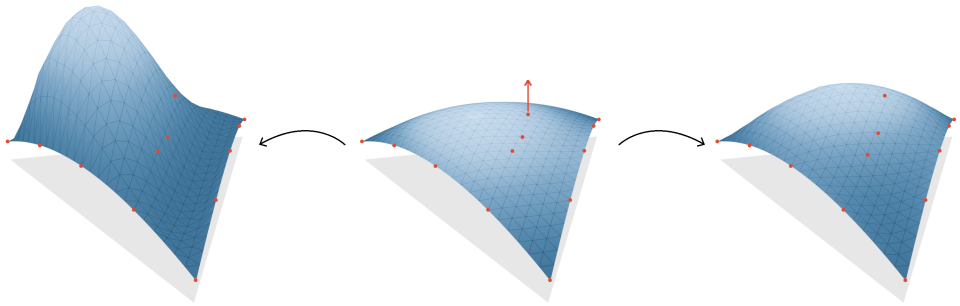


Figure 9.8. Impact of increasing the z-value of q_{112} when the central interpolation nodes are close to collinear (middle). Insisting on the interpolation at q_{121} and q_{211} and using the displacement vectors Δ_1^p to update the Bézier points leads to a big change of the surface shape (left). A better result is obtained by utilizing the blending technique (9.22), even though this modification comes at the price of slightly perturbing q_{121} and q_{211} (right).

where E is defined in (9.19). We blend this with the result obtained with the least-squares method. Let $\widehat{\boldsymbol{\lambda}}_{112}$ be the projection of $\boldsymbol{\lambda}_{112}$ onto L_{112}. Let $t = \frac{4\sqrt{6}}{3}h$ where h is the distance $h = \|\widehat{\boldsymbol{\lambda}}_{112} - \boldsymbol{\lambda}_{112}\|$. We define a blending function

$$f(t) = \begin{cases} 0, & \text{if } t > 1, \\ (1-t)^2(1+2t), & \text{if } t \in [0,1]. \end{cases} \tag{9.21}$$

We assume that $\boldsymbol{\lambda}_{\widehat{\boldsymbol{\lambda}}_{112}} = \widehat{\Delta}_{112}^q = \Delta_{112}^q f(t)$. In geometric terms, we propose that the displacement of q_{112} by Δ_{112}^q results in a proportional displacement of the surface point $p(\widehat{\boldsymbol{\lambda}}_{112})$ by $\Delta_{112}^q f(t)$. Let $\widehat{V'}$ be the Vandermonde matrix (9.20) applied to $\widehat{\boldsymbol{\lambda}}_{112}, \boldsymbol{\lambda}_{211}, \boldsymbol{\lambda}_{121}$. We obtain a least-square solution $\widehat{\Delta}^p$ as in Proposition 9.4,

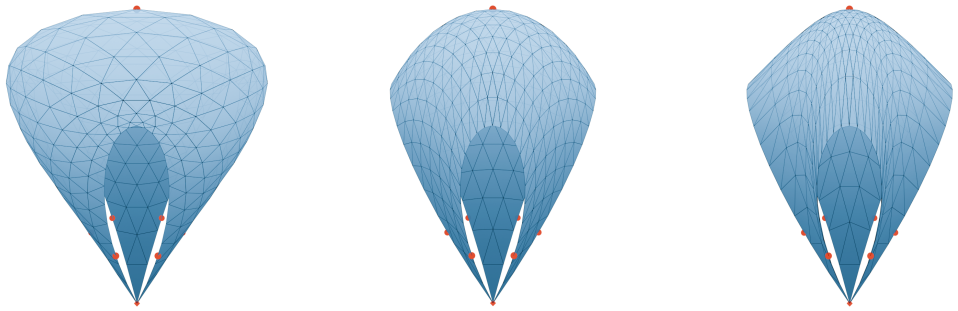
$$\widehat{\Delta}^p = \frac{1}{\mu_1^2 + \mu_2^2} \begin{pmatrix} \nu_2^T \\ \nu_3^T \end{pmatrix}^\dagger \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \Delta_{112}^q f(t),$$

where μ_1 and μ_2 are as above. Now the remaining issue is $\nu_1^T \widehat{\Delta}^p \neq \Delta_{112}^q$. We employ a technique in [8] to obtain a vector Δ_2^p such that $\nu_1^T \Delta_2^p = \Delta_{112}^q$, namely

$$\Delta_2^p = \widehat{\Delta}^p + \frac{\nu_1}{\nu_1^T \nu_1} (\Delta_{112}^q - \nu_1^T \widehat{\Delta}^p).$$

We then get the displacement vectors as the convex combination

$$\Delta^p = \Delta_1^p (1 - f(t)) + f(t) \Delta_2^p. \tag{9.22}$$



$\beta_{111} = 0.3$ $\beta_{111} = 1$ $\beta_{111} = 2$

Figure 9.9. Effect of changing the weight β_{111} of the central surface point q_{111} (the point on top) of a rational cubic surface.

The effect of this blend can be seen in Figure 9.8. We note that employing these least squares solutions necessitates resampling the central points, which can be efficiently computed as

$$\widetilde{q}_{ijk} = q_{ijk} + v_{ijk}\Delta^p,$$

where v_{ijk} is the row vector corresponding to $\boldsymbol{\lambda}_{ijk}$ in the Vandermonde matrix V'.

9.5 Rational case and effect of changing the weights

In the rational setting, the weights β_{ijk} give additional degrees of freedom. Hence, it is natural to question their role, that is, to describe geometrically the effect of manipulating them. It turns out that, by varying a weight β_{ijk}, we observe that the flatness of the surface around q_{ijk} can be adjusted accordingly, as shown in Figure 9.9. This phenomenon is consistent with similar rational (quadratic) triangular surfaces [114]. By decreasing β_{ijk}, we flatten the surface in the vicinity of q_{ijk}, and by increasing β_{ijk}, we increase the local curvature.

Proposition 9.5. *Consider the rational interpolation* (9.6). *The Jacobian* $J_{\boldsymbol{b}}(\boldsymbol{\lambda}_{abc})$ *is given by*

$$J_{\boldsymbol{b}}(\boldsymbol{\lambda}_{abc}) = \frac{1}{\beta_{abc}} \sum_{(i,j,k)\in\mathscr{I}_n\setminus\{(a,b,c)\}} \beta_{ijk}\nabla\ell_{ijk}(\boldsymbol{\lambda}_{abc})(q_{ijk} - q_{abc})^T. \tag{9.23}$$

Proof. It follows from (9.6) that

$$\sum_{(i,j,k)\in\mathscr{I}_n} \ell_{ijk}(\boldsymbol{u})\beta_{ijk}(q_{ijk} - p(\boldsymbol{u})) = 0.$$

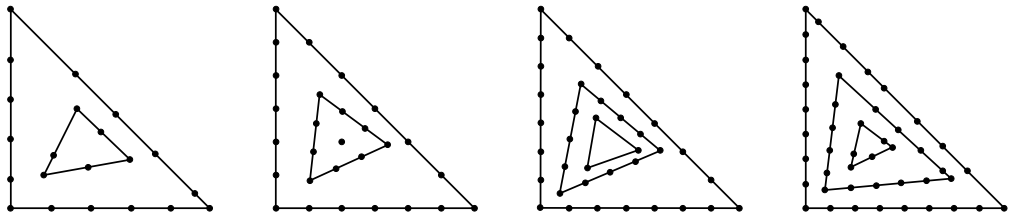


Figure 9.10. Examples of interpolation node configurations with non-singular Vandermonde matrices for $n = 5, 6, 7, 8$.

Since $\boldsymbol{u} = (u, v, w)$ with $u + v + w = 1$, we can consider $p(\boldsymbol{u})$ as a bivariate function. Hence, we have

$$\sum_{(i,j,k)\in\mathscr{I}_n} \nabla\ell_{ijk}(\boldsymbol{u})\beta_{ijk}(q_{ijk} - p(\boldsymbol{u}))^T - J_{\boldsymbol{b}}(\boldsymbol{u}) \sum_{(i,j,k)\in\mathscr{I}_n} \ell_{ijk}(\boldsymbol{u})\beta_{ijk} = 0.$$

Now substituting $\boldsymbol{u} = \boldsymbol{\lambda}_{abc}$, we easily deduce (9.23). □

The term in the summation in (9.23) does not depend on β_{abc}, thus confirming our observation.

9.6 High degree

While conditions GC1, CG3, and GC4 describe these configurations for surfaces of degree $n = 2, 3, 4$, respectively, a natural extension of these conditions to higher degree is to arrange the interpolation nodes such that they form a set of nested triangles $T_1, \ldots, T_{\lfloor (n+3)/3 \rfloor}$, with $n + 4 - 3k$ nodes on each side of T_k (see Figure 9.10). Like in the proofs of Propositions 9.1 and 9.3, Bezout's theorem guarantees the non-singularity of the Vandermonde matrix for these configurations.

Chapter 10

Conclusion

In this thesis, we focus on shape control tools for curve and surface design. In general, we want to use Bézier curves. They have elegant geometric meaning and efficient evaluation algorithms. We obtain an additional degree-of-freedom through weights. Now, a curve can be manipulated by repositioning the control points or by tweaking the weights. Manipulating a point individually is very intuitive and has a pseudo-local effect. The points can also be manipulated simultaneously to achieve single-point interpolation or to tweak the local curvature. The weights can be individually manipulated with the help of auxiliary weight points or manipulated simultaneously through shape factors represented geometrically by the Farin points and the shoulder points. Manipulation of a weight w_k individually induces a push / pull effect on the curves following a projection with the centre p_k. Farin points and shoulder points provide more intuitive and pseudo-local controls. They differ in the fact that the Farin points vary under Möbius transformations. Since the Farin point F_k belongs to the segment $[p_k, p_{k+1}]$, intuitive manipulation through F_k becomes challenging if the consecutive points p_k and p_{k+1} are very close to each other. Similarly, a shoulder point S_k belongs to a segment $[p_k, r_k]$ where r_k is the midpoint of the segment $[p_{k-1}, p_{k+1}]$. Hence, manipulating S_k becomes challenging when the points p_{k-1}, p_k, p_{k+1} are close to being collinear. Polynomial Bézier curves can also be manipulated via its Gauss–Legendre control polygon.

We further obtain more degrees of freedom by expressing a rational curve in barycentric form. We can snap a degree n curve to $n+1$ points. The shape of the interpolating basis functions is not as well-shaped as the Bernstein polynomials. Hence, when manipulating an interpolation point q_k, we propose that if one prefers to keep the nodes fixed, then the manipulation should only be for micro-editing. Alternatively, if the nodes can be flexible, then we propose

a technique that allows large manipulation of the control points that is similar to cubic spline manipulation. Interpolation points can be repositioned to snap curves onto specific points. If one wants to adjust a part of a curve where there is no interpolation point, one can slide an existing interpolation to that place, or one can simply insert new ones. Similar shape controls are explored for periodic rational Bézier curves. In the periodic case, the manipulation of curves given by Floater–Hormann interpolation with respect to adaptive parameters can be studied to allow for large manipulation of the interpolation points.

Rational Bézier curves are also popular due to their fast evaluation time. We studied the efficiency of the evaluation of rational Bézier curves. We observe that the method involving conversion to the barycentric form with respect to the Chebyshev nodes improves the existing algorithms.

The representation of trigonometric polynomial curves as type-1 and type-2 tangent interpolating curves that we propose enriches the existing modelling capabilities of these curves by providing novel shape control tools. While trigonometric polynomials are a natural space for modelling closed curves, the main drawback so far was that neither the Bézier nor the Lagrange representation offers intuitive shape control for complex curves with a large number of control points. This limitation is now remedied to a large extent by our type-2 representation, which establishes a relation between the control polygon and the shape of the curve that is very close to the corresponding relation in the case of cubic B-spline curves, regardless of the number of control points. Moreover, it is well known in the case of classical Bézier and B-spline curves that rational curves can model a bigger set of shapes. Hence, it could be worthwhile to investigate how periodic rational Bézier curves can be expressed in tangent interpolating form with additional shape parameters. Finally, it is also well known that, for B-spline curves, uniform nodes are not the proper choice if the control edge lengths vary a lot. Akin to the B-spline setting, future work should therefore explore the construction of trigonometric tangent interpolating curves with respect to nonuniform nodes.

Finally, we describe a geometric configuration of the nodes so that triangular patches can be expressed in an interpolating form. Since we are interested in geometric design, having these geometric descriptions of the nodes is important. We observe that the Lagrange interpolation exists if the nodes belong to a set of triangles with a specific number of points on each edge. For low-degree $n = 2, 3, 4$, these configurations are flexible enough for intuitive design. For $n = 4$, we develop a technique that allows the inner triangle to be degenerate and still have an intuitive manipulation.

Appendix A

Algorithms

A.1 de Casteljau algorithm

Algorithm 2 toHomogeneous($P_0, \ldots, P_n, w_0, \ldots, w_n$)

for $k \leftarrow 0(1)n$ **do**
  $\widehat{P}_k \leftarrow (w_k P_k, w_k)$
end for
return $\widehat{P}_0, \ldots, \widehat{P}_n$

Algorithm 3 deCasteljau($P_0, \ldots, P_n, t$)

for $k \leftarrow 0(1)n$ **do**
  $\widehat{P}_k \leftarrow P_k$
end for
$t_1 \leftarrow 1 - t$
for $r \leftarrow 1(1)n$ **do**
  for $k \leftarrow 0(1)(n-r)$ **do**
    $\widehat{P}_k \leftarrow \widehat{P}_k t_1 + t\widehat{P}_{k+1}$.
  end for
end for
$P \leftarrow \text{proj}(\widehat{P}_0)$ ▷ central projection
return P

Algorithm 4 RationalDeCasteljau($P_0, \ldots, P_n, w_0, \ldots, w_n, t$)

$t_1 \leftarrow 1 - t$
for $r \leftarrow 1(1)n$ **do**
  for $k \leftarrow 0(1)(n-r)$ **do**
    $u \leftarrow t_1 w_k$
    $v \leftarrow t w_{k+1}$
    $w_k \leftarrow u + v$
    $c_1 \leftarrow u / w_k$
    $c_2 \leftarrow 1 - c_1$
    $P_k \leftarrow P_k c_1 + c_2 P_{k+1}$.
  end for
end for
return P_0

A.2 VS algorithm

Algorithm 5 Preprocessing_of_VS_and_HornBez($P_0, \ldots, P_n, w_0, \ldots, w_n$)

$b \leftarrow 1$
$P_0 \leftarrow w_0 P_0$
$P_n \leftarrow w_n P_n$
for $k = 1(1)(n-1)$ **do**
  $b \leftarrow b(n+1-k)$
  $b \leftarrow b/k$
  $w_k \leftarrow b w_k$
  $P_k \leftarrow w_k P_k$
end for
return $(P_0, \ldots, P_n, w_0, \ldots, w_n)$

Algorithm 6 VS($P_0, \ldots, P_n, w_0, \ldots, w_n, t$)

if $t \leq 1/2$ **then**
  $s \leftarrow t/(1-t)$
  $d \leftarrow w_n$
  $N \leftarrow P_n$
  for $k = 1(1)n$ **do**
    $n_k \leftarrow n-k$
    $N \leftarrow Ns + P_{n_k}$
    $d \leftarrow ds + w_{n_k}$
  end for
else if $t > 1/2$ **then**
  $s \leftarrow (1-t)/t$
  $d \leftarrow w_0$
  $N \leftarrow P_0$
  for $k = 1(1)n$ **do**
    $N \leftarrow Ns + P_k$
    $d \leftarrow ds + w_k$
  end for
end if
return N/d

A.3 Horner Bézier algorithm

Algorithm 7 HornBez($P_0, \ldots, P_n, w_0, \ldots, w_n, t$)

$s \leftarrow 1-t$
$t_k \leftarrow 1$
$d \leftarrow sw_0$
$N \leftarrow sP_0$
for $k \leftarrow 1(1)(n-1)$ **do**
  $t_k \leftarrow t_k t$
  $N \leftarrow (N + t_k P_k)s$
  $d \leftarrow (d + t_k w_k)s$
end for
$t_k \leftarrow t_k t$
$N \leftarrow N + t_k P_n$
$d \leftarrow d + t_k w_n$
return N/d

A.4 Geometric algorithm

Algorithm 8 Geometric$(P_0, \ldots, P_n, w_0, \ldots, w_n, t)$

$h \leftarrow 1$
$u \leftarrow 1 - t$
$n_1 \leftarrow n + 1$
$N \leftarrow P_0$
if $t \leq 1/2$ **then**
  $u \leftarrow t/u$
  for $k \leftarrow 1(1)n$ **do**
    $h \leftarrow hu(n_1 - k)w_k$
    $h \leftarrow h/(kw_{k-1} + h)$
    $h_1 \leftarrow 1 - h$
    $N \leftarrow h_1 N + hP_k$
  end for
else if $t > 1/2$ **then**
  $u \leftarrow u/t$
  for $k \leftarrow 1(1)n$ **do**
    $h \leftarrow h(n_1 - k)w_k$
    $h \leftarrow h/(kuw_{k-1} + h)$
    $h_1 \leftarrow 1 - h$
    $N \leftarrow h_1 N + hP_k$
  end for
end if
return N

A.5 Wang–Ball algorithm

Algorithm 9 AC_coefficients(n)

$p_2 \leftarrow 1$
$M \leftarrow \mathbf{1}_{n+1}$ ▷ identity matrix of size $(n+1) \times (n+1)$
for $i \leftarrow 0(1)(\lceil n/2 \rceil - 1)$ **do**
  $N_i \leftarrow n-2-2i$
  for $k \leftarrow 0(1)n$ **do**
    if $i < k$ and $N_i \geq 0$ **then**
      $M_{k,i} \leftarrow M_{k-1,i}(N_i - k + i + 1)$
      $M_{k,i} \leftarrow M_{k,i}/(k-i)$
    end if
  end for
  for $k \leftarrow 0(1)n$ **do**
    $M_{k,i} \leftarrow M_{k,i} p_2$
    $M_{n-k,n-i} \leftarrow M_{k,i}$
  end for
  $p_2 \leftarrow 2p_2$
end for
return M

Algorithm 10 toWangBall($P_0, \ldots, P_n, w_0, \ldots, w_n$)

$(\widehat{P}_0, \ldots, \widehat{P}_n) \leftarrow$ toHomogeneous$(P_0, \ldots, P_n, w_0, \ldots, w_n)$
for $i \leftarrow 0(1)n$ **do**
  $\widehat{R}_i \leftarrow 0$
end for
$b \leftarrow 1$
$c \leftarrow 1$
$\widehat{R}_0 \leftarrow \widehat{P}_0$
$\widehat{R}_n \leftarrow \widehat{P}_n$
$M \leftarrow$AC_coefficients(n)
$k \leftarrow 1$
while $k \leq n-k$ **do**
  $b \leftarrow b(n-k+1)/k$
  $c \leftarrow c/2$
  $\widehat{R}_k \leftarrow (b\widehat{P}_k - \langle M_k, (\widehat{R}_0, \ldots, \widehat{R}_n)\rangle)c$ ▷ $\langle a, b\rangle$ is a scalar product
  if $k = n-k$ **then**
    stop
  end if
  $K \leftarrow n-k$
  $\widehat{R}_K \leftarrow (b\widehat{P}_K - \langle M_K, (\widehat{R}_0, \ldots, \widehat{R}_n)\rangle)c$
  $k \leftarrow k+1$
end while
for $i \leftarrow 0(1)n$ **do**
  $R_i \leftarrow \text{proj}(\widehat{R}_i)$
  $v_i \leftarrow \widehat{R}_i^z$ ▷ z-coordinate of $\widehat{R}_i$
end for
return $(R_0, \ldots, R_n, v_0, \ldots, v_n)$

Algorithm 11 WangBall($R_0, \ldots, R_n, v_0, \ldots, v_n, t$)

$k \leftarrow n$
$J \leftarrow 0$
$s \leftarrow 1 - t$
while $k > 2$ **do**
  if k is odd **then**
    $k_1 \leftarrow (k-1)/2$
    $k_2 \leftarrow (k+1)/2$
    $a \leftarrow s v_{k_1}$
    $b \leftarrow t v_{k_2}$
    $v_{k_1} \leftarrow a + b$
    $R_{k_1} \leftarrow (R_{k_1} a + b R_{k_2})/v_{k_1}$
    if $J = 0$ **then**
      $J \leftarrow k_2 + 1$
    end if
  else
    $k_2 \leftarrow k/2$
    if $J = 0$ **then**
      $J \leftarrow k_2 + 1$
    end if
    $a \leftarrow s v_{k_2-1}$
    $b \leftarrow t v_{k_2}$
    $v_{k_2-1} \leftarrow a + b$
    $R_{k_2-1} \leftarrow (R_{k_2-1} a + b R_{k_2})/v_{k_2-1}$
    $a \leftarrow s v_{k_2}$
    $b \leftarrow t v_J$
    $v_{k_2} \leftarrow a + b$
    $R_{k_2} \leftarrow (R_{k_2} a + b R_J)/v_{k_2}$
    $J \leftarrow J + 1$
  end if
  $k \leftarrow k - 1$
end while
$a \leftarrow s v_0$
$b \leftarrow t v_1$
$c \leftarrow s v_1$
$d \leftarrow t v_n$
$w_q \leftarrow a + b$
$w_r \leftarrow c + d$
$e \leftarrow s w_q$
$f \leftarrow t w_r$
$w_w \leftarrow e + f$
$Q \leftarrow (R_0 a + b R_1)/w_q$
$V \leftarrow (R_1 c + d R_n)/w_r$
$R \leftarrow (Q e + f V)/w_w$
return R

A.6 Bernstein–Fourier algorithm

Algorithm 12 RealProduct(u, v)

$a \leftarrow \mathrm{Re}(u)\,\mathrm{Re}(v)$
$b \leftarrow \mathrm{Im}(u)\,\mathrm{Im}(v)$
return $a - b$

Algorithm 13 ToHomogeneousAndIfft$(P_0, \ldots, P_n, \nu_0, \ldots, \nu_n)$

for $i \leftarrow 0(1)n$ **do**
    $\widehat{S}_i \leftarrow (\nu_i P_i, \nu_i)$
end for
$(S_0, \ldots, S_n) \leftarrow \mathrm{ifft}(S_0, \ldots, S_n)$
return $(S_0, \ldots, S_n)$

Algorithm 14 BernsteinFourier($S_0, \ldots, S_n, \zeta_1, \ldots, \zeta_n, t$)

$p \leftarrow 0$
$t_1 \leftarrow 1 - t$
if n is even **then**
  $N \leftarrow n/2 + 1$
else
  $N \leftarrow (n+1)/2$
end if
for $i \leftarrow 1(1)(N-1)$ **do**
  $u \leftarrow \zeta_i t + t_1$
  $u \leftarrow \text{pow}(u, n)$
  $p \leftarrow p + \text{RealProduct}(u, Q_i)$
end for
$p \leftarrow 2p + Q_0$
if n is odd **then**
  $u \leftarrow 1 - 2t$
  $u \leftarrow \text{pow}(u, n)$
  $p \leftarrow p + uQ_N$
end if
return proj(p)

Algorithm 15 BernsteinFourier_2($S_0, \ldots, S_n, \zeta_1, \ldots, \zeta_n, t$)

$p_t \leftarrow 0$
$p_s \leftarrow 0$
$t_1 \leftarrow 1 - t$
if n is even **then**
  $N \leftarrow n/2 + 1$
else
  $N \leftarrow (n+1)/2$
end if
for $i \leftarrow 1(1)(N-1)$ **do**
  $u \leftarrow \zeta_i t + t_1$
  $u \leftarrow \text{pow}(u, n)$
  $p_t \leftarrow p_t + \text{RealProduct}(u, Q_i)$
  $u \leftarrow \overline{u}/\omega_i$
  $p_s \leftarrow p_s + \text{RealProduct}(u, Q_i)$
end for
$p_t \leftarrow 2p_t + Q_0$
$p_s \leftarrow 2p_s + Q_0$
if n is odd **then**
  $u \leftarrow 1 - 2t$
  $u \leftarrow \text{pow}(u, n)$
  $p_t \leftarrow p_t + uQ_N$
  $p_s \leftarrow p_s - uQ_N$
end if
return $(\text{proj}(p_t), \text{proj}(p_s))$

A.7 Barycentric form

Algorithm 16 AdaptedVS($P_0, \ldots, P_n, w_0, \ldots, w_n, t$)

$c \leftarrow 1$
if $t \leq 1/2$ **then**
  $t_1 \leftarrow 1 - t$
  $s \leftarrow t/t_1$
  $c \leftarrow \text{pow}(t_1, n)$
  $d \leftarrow \omega_n$
  $N \leftarrow P_n$
  for $k = 1(1)n$ **do**
    $n_k \leftarrow n - k$
    $N \leftarrow Ns + P_{n_k}$
    $d \leftarrow ds + w_{n_k}$
  end for
else if $t > 1/2$ **then**
  $s \leftarrow (1-t)/t$
  $c \leftarrow \text{pow}(t, n)$
  $d \leftarrow \omega_0$
  $N \leftarrow P_0$
  for $k = 1(1)n$ **do**
    $N \leftarrow Ns + P_k$
    $d \leftarrow ds + w_k$
  end for
end if
return $\{N/d, cd\}$

Algorithm 17 ToBarycentric$(P_0,\ldots,P_n,w_0,\ldots,w_n)$

$b \leftarrow 1$
sgn $\leftarrow 1$
$(P_0,\ldots,P_n,w_0,\ldots,w_n) \leftarrow$Preprocessing_of_VS_and_HornBez$(P_0,\ldots,P_n,w_0,\ldots,w_n)$
for $k = 0(1)n$ **do**
  if UNIFORM **then**
    $t_k \leftarrow k/n$
    $Q_k, z \leftarrow$ AdaptedVS$(P_0,\ldots,P_n,w_0,\ldots,w_n,t_k)$
    $u_k \leftarrow \operatorname{sgn} bz$
    $b \leftarrow b(n+1-(k+1))$
    $b \leftarrow b/(k+1)$
  else if CHEBYSHEV **then**
    $t_k \leftarrow \cos(k\pi/n)$
    $Q_k, z \leftarrow$ AdaptedVS$(P_0,\ldots,P_n,w_0,\ldots,w_n,t_k)$
    $b \leftarrow 1$
    if $k = 0$ or $k = n$ **then**
      $b \leftarrow 0.5$
    end if
    $u_k \leftarrow \operatorname{sgn} bz$
  end if
  sgn $\leftarrow -$sgn
end for
return $Q_0,\ldots,Q_n,u_0,\ldots,u_n,t_0,\ldots,t_n$

Algorithm 18 Barycentric$(Q_0,\ldots,Q_n,u_0,\ldots,u_n,t_0,\ldots,t_n,t)$

$Q \leftarrow 0$
$d \leftarrow 0$
for $k \leftarrow 0(1)n$ **do**
  $u \leftarrow t - t_k$
  if $u = 0$ **then return** Q_k
  end if
  $u \leftarrow u_k/u$
  $Q \leftarrow Q + uQ_k$
  $d \leftarrow d + u$
end for
$Q \leftarrow Q/d$
return Q

Algorithm 19 Barycentric2$(Q_0, \ldots, Q_n, u_0, \ldots, u_n, t_0, \ldots, t_n, t)$

$Q_1 \leftarrow 0$
$d_1 \leftarrow 0$
$Q_2 \leftarrow 0$
$d_2 \leftarrow 0$
for $k \leftarrow 0(1)n$ **do**
  $u \leftarrow t - t_k$
  $n_k \leftarrow n - k$
  if $u = 0$ **then return** $\{Q_k, Q_{n_k}\}$
  end if
  $v \leftarrow u_{n_k}/u$
  $u \leftarrow u_k/u$
  $Q_1 \leftarrow Q_1 + uQ_k$
  $d_1 \leftarrow d_1 + u$
  $Q_2 \leftarrow Q_2 + vQ_{n_k}$
  $d_2 \leftarrow d_2 + v$
end for
$Q1 \leftarrow Q1/d1$
$Q2 \leftarrow Q2/d2$
return $\{Q_1, Q_2\}$

Bibliography

[1] L. Adams. 1952. The use of conics in airplane design. *Mathematics Magazine* 25, 4 (1952), 195–202. doi:10.2307/3029785

[2] R. Ait-Haddou, T. Nomura, and L. Biard. 2010. A refinement of the variation diminishing property of Bézier curves. *Computer Aided Geometric Design* 27, 2 (2010), 202–211. doi:10.1016/j.cagd.2009.12.001

[3] A. Antoulas and B. Anderson. 1986. On the scalar rational interpolation problem. *IMA Journal of Mathematical Control and Information* 3, 2–3 (1986), 61–88. doi:10.1093/imamci/3.2-3.61

[4] M. Artin. 2011. *Algebra*. Pearson Education.

[5] T. Banchoff and S. Lovett. 2022. *Differential Geometry of Curves and Surfaces*. CRC Press.

[6] P. Barry and R. Goldman. 1988. A recursive evaluation algorithm for a class of Catmull–Rom splines. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '88)*. Association for Computing Machinery, New York, NY, USA, 199–204. doi:10.1145/54852.378511

[7] B. Barsky. 1986. *Arbitrary Subdivision of Bézier Curves*. Technical Report UCB/CSD-86-265. EECS Department, University of California, Berkeley.

[8] R. Bartels and J. Beatty. 1989. A technique for the direct manipulation of spline curves. In *Proceedings of the Graphics Interface '89* (1989). 33–39. doi:10.20380/GI1989.06

[9] S. Bernstein. 1912. Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités. *Communications of the Mathematical Society of Kharkov* 13 (1912), 1–2.

[10] J.-P. Berrut. 1984. Baryzentrische formeln zur trigonometrischen interpolation (I). *Zeitschrift für angewandte Mathematik und Physik* 35, 1 (jan 1984), 91–105. doi:10.1007/BF00945179

[11] J.-P. Berrut. 1988. Rational functions for guaranteed and experimentally well-conditioned global interpolation. *Computers & Mathematics with Applications* 15, 1 (1988). doi:10.1016/0898-1221(88)90067-3

[12] J.-P. Berrut and H. Mittelmann. 1997. Lebesgue constant minimizing linear rational interpolation of continuous functions over the interval. *Computers & Mathematics with Applications* 33, 6 (1997), 77–86. doi:10.1016/S0898-1221(97)00034-5

[13] J.-P. Berrut and L. Trefethen. 2004. Barycentric Lagrange Interpolation. *SIAM Rev.* 46, 3 (Sept. 2004). doi:10.1137/S0036144502417715

[14] L. Bezerra. 2012. Vandermonde factorizations of a regular Hankel matrix and their application to the computation of Bézier curves. *SIAM J. Matrix Anal. Appl.* 33, 2 (2012), 411–432. doi:10.1137/100800300

[15] L. Bezerra. 2013. Efficient computation of Bézier curves from their Bernstein–Fourier representation. *Appl. Math. Comput.* 220 (2013), 235–238. doi:10.1016/j.amc.2013.05.079

[16] L. Bezerra and L. Sacht. 2011. On computing Bézier curves by Pascal matrix methods. *Appl. Math. Comput.* 217, 24 (2011), 10118–10128. doi:10.1016/j.amc.2011.05.007

[17] P. Bézier. 1966. Définition numérique des courbes et surfaces. *Automatisme* 11 (1966), 625–632.

[18] P. Bézier, W. Hawthorne, and G. Edwards. 1971. Example of an existing system in the motor industry: the Unisurf system. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 321, 1545 (1971), 207–218. doi:10.1098/rspa.1971.0027

[19] M. Blyth and C. Pozrikidis. 2005. A Lobatto interpolation grid over the triangle. *IMA Journal of Applied Mathematics* 71 (03 2005), 1–17. doi:10.1093/imamat/hxh077

[20] W. Boehm and A. Müller. 1999. On de Casteljau's algorithm. *Computer Aided Geometric Design* 16, 7 (1999), 587–605. doi:10.1016/S0167-8396(99)00023-0

[21] L. Bos. 1983. Bounding the Lebesgue function for Lagrange interpolation in a simplex. *Journal of Approximation Theory* 38, 1 (1983), 43–59. doi:10.1016/0021-9045(83)90140-5

[22] L. Bos, S. De Marchi, K. Hormann, and G. Klein. 2012. On the Lebesgue Constant of Barycentric Rational Interpolation at Equidistant Nodes. *Numer. Math.* 121, 3 (2012), 461–471. doi:10.1007/s00211-011-0442-8

[23] S. Boyd and L. Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

[24] M. Briani, A. Sommariva, and M. Vianello. 2012. Computing Fekete and Lebesgue points: Simplex, square, disk. *J. Comput. Appl. Math.* 236, 9 (2012), 2477–2486. doi:10.1016/j.cam.2011.12.006

[25] G. Brunnett, T. Schreiber, and J. Braun. 1996. The geometry of optimal degree reduction of Bézier curves. *Computer Aided Geometric Design* 13, 8 (1996), 773–788. doi:10.1016/0167-8396(96)00009-X

[26] J. Carnicer. 1999. Interpolation, shape control and shape properties. In *Shape Preserving Representations in Computer-Aided Geometric Design*, Juan M. Peña (Ed.). Nova Science Publishers, Commack, Chapter 2, 15–43.

[27] B. Casselman. 2005. *Mathematical Illustrations: A Manual of Geometry and PostScript*. Cambridge University Press.

[28] E. Catmull and R. Rom. 1974. A class of local interoplating splines,. In *Computer Aided Geometric Design*, Robert Barnhill and Richard Riesenfeld (Eds.). Academic Press, 317–326. doi:10.1016/B978-0-12-079050-0.50020-5

[29] A.-L. Cauchy. 1815. Mémoire sur les fonctions qui ne peuvent obtenir que deux valeurs égales et de signes contraires par suite des transpositions opérées entre les variables qu'elles renferment. *Journal de l'École polytechnique* 10, 17 (1815), 91–169. https://gallica.bnf.fr/ark:/12148/bpt6k90193x Reprinted in Œuvres complètes, série 2, tome 1.

[30] Q. Chen and I. Babuška. 1995. Approximate optimal points for polynomial interpolation of real functions in an interval and in a

triangle. *Computer Methods in Applied Mechanics and Engineering* 128, 3 (1995), 405–417. doi:10.1016/0045-7825(95)00889-6

[31] J. Chou. 1995. Higher order Bézier circles. *Computer-Aided Design* 27, 4 (1995), 303–309. doi:10.1016/0010-4485(95)91140-G

[32] K. Chung and T. Yao. 1977. On Lattices Admitting Unique Lagrange Interpolations. *SIAM J. Numer. Anal.* 14, 4 (1977), 735–743. doi:10.1137/0714050

[33] S. Conte and C. de Boor. 2017. *Elementary Numerical Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611975208

[34] C. de Boor. 2005. Divided Differences. *Surveys in Approximation Theory* 1 (02 2005), 1–23. doi:10.48550/arXiv.math/0502036

[35] N. Dejdumrong. 2006. Rational DP-Ball Curves. In *International Conference on Computer Graphics, Imaging and Visualisation (CGIV'06)*. 478–483. doi:10.1109/CGIV.2006.74

[36] N. Dejdumrong. 2008. Efficient Algorithms for Non-Rational and Rational Bézier Curves. In *International Conference on Computer Graphics, Imaging and Visualisation (CGIV'08)*. 109–114. doi:10.1109/CGIV.2008.62

[37] N. Dejdumrong, H. Phien, H. Tien, and K. Lay. 2001. Rational Wang–Ball curves. *International Journal of Mathematical Education in Science and Technology* 32, 4 (2001), 565–584. doi:10.1080/00207390110038358

[38] J. Delgado and J. Peña. 2003. A shape preserving representation with an evaluation algorithm of linear complexity. *Computer Aided Geometric Design* 20, 1 (2003), 1–10. doi:10.1016/S0167-8396(02)00190-5

[39] Jorge Delgado and Juan Manuel Peña. 2004. A Shape Preserving Representation for Rational Curves with Efficient Evaluation Algorithm. In *Advances in Geometric Modeling*. John Wiley & Sons, Ltd, Chapter 3, 39–54. doi:10.1002/0470860448.ch3

[40] M. Dupuy. 1948. Le calcul numérique des fonctions par l'interpolation barycentrique. *Comptes Rendus de l'Académie des Sciences* 226 (1948), 158–159.

[41] M. Eck. 1993. Degree reduction of Bézier curves. *Computer Aided Geometric Design* 10, 3 (1993), 237–251. doi:10.1016/0167-8396(93)90039-6

[42] J.-J. Fang and C.-L. Hung. 2013. An improved parameterization method for B-spline curve and surface interpolation. *Computer-Aided Design* 45, 6 (2013), 1005–1028. doi:10.1016/j.cad.2013.01.005

[43] G. Farin. 1983. Algorithms for rational Bézier curves. *Computer-Aided Design* 15, 2 (1983), 73–77. doi:10.1016/0010-4485(83)90171-9

[44] G. Farin. 1999. *NURBS for Curve & Surface Design: From Projective Geometry to Practical Use*. CRC Press.

[45] G. Farin. 2001. *Curves and Surfaces for CAGD: A Practical Guide* (5th ed.). Morgan Kaufmann, San Francisco. doi:10.1016/B978-1-55860-737-8.X5000-5

[46] G. Farin and D. Hansford. 2000. *The Essentials of CAGD* (1th ed.). A K Peters/CRC Press, New York.

[47] G. Farin and A. Worsey. 1991. Reparametrization and degree elevation of rational Bézier curves. In *NURBS for Curve and Surface Design*, Gerald Farin (Ed.). Society for Industrial and Applied Mathematics, 47–58.

[48] R. Farouki. 2012. The Bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design* 29, 6 (2012), 379–419. doi:10.1016/j.cagd.2012.03.001

[49] M. Floater and K. Hormann. 2007. Barycentric rational interpolation with no poles and high rates of approximation. *Numer. Math.* 107, 2 (2007), 315–331. doi:10.1007/s00211-007-0093-y

[50] T. Foley and G. Nielson. 1989. Knot Selection for Parametric Spline Interpolation. In *Mathematical Methods in Computer Aided Geometric Design*, TOM LYCHE and LARRY L. SCHUMAKER (Eds.). Academic Press, 261–CP4. doi:10.1016/B978-0-12-460515-2.50023-8

[51] A. Forrest. 1990. Interactive interpolation and approximation by Bézier polynomials. *Computer-Aided Design* 22, 9 (1990), 527–537. doi:10.1016/0010-4485(90)90038-E

[52] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. 2007. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions of Mathematical Software* 33, 2, Article 13 (June 2007), 15 pages. doi:10.1145/1236463.1236468

[53] C. Fuda, A. Ramanantoanina, and K. Hormann. 2024. A comprehensive comparison of algorithms for evaluating rational Bézier curves. *Dolomites Research Notes on Approximation* 17, 3 (Sept. 2024), 56–79. doi:10.14658/PUPJ-DRNA-2024-3-9

[54] W. Fulton. 1969. *Algebraic Curves: An Introduction to Algebraic Geometry*. Addison-Wesley Publishing Company, Advanced Book Program.

[55] F. Gantmacher. 1959. *The Theory of Matrices*. Vol. 2. Chelsea, New York.

[56] Q. Ge, L. Srinivasan, and J. Rastegar. 1997. Low-harmonic rational Bézier curves for trajectory generation of high-speed machinery. *Computer Aided Geometric Design* 14, 3 (1997), 251–271. doi:10.1016/S0167-8396(96)00032-5

[57] R. Goldman. 2003. *Pyramid Algorithms*. Morgan Kaufmann, San Francisco. 187–306 pages. doi:10.1016/B978-1-55860-354-7.X5000-4

[58] I. Gradshteyn and I. Ryzhik. 2007. *Table of Integrals, Series, and Products* (7 ed.). Academic Press.

[59] G. Guennebaud, B. Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

[60] D. Hansford. 2002. Chapter 4 - Bézier Techniques. In *Handbook of Computer Aided Geometric Design*, Gerald Farin, Josef Hoschek, and Myung-Soo Kim (Eds.). North-Holland, Amsterdam, 75–109. doi:10.1016/B978-044451104-1/50005-8

[61] P. Hartley and C. Judd. 1978. Parametrization of Bézier-type B-spline curves and surfaces. *Computer-Aided Design* 10, 2 (1978), 130–134. doi:10.1016/0010-4485(78)90090-8

[62] P. Henrici. 1965. *Elements of Numerical Analysis*. John Wiley and Sons, New York, 1964. 183–262 pages.

[63] P. Henrici. 1979. Barycentric formulas for interpolating trigonometric polynomials and their conjugates. *Numer. Math.* 33, 2 (jun 1979). doi:10.1007/BF01399556

[64] P. Henrici. 1979. Fast Fourier Methods in Computational Complex Analysis. *SIAM Rev.* 21, 4 (1979), 481–527. doi:10.1137/1021093

[65] J. Hesthaven. 1998. From Electrostatics to Almost Optimal Nodal Sets for Polynomial Interpolation in a Simplex. *SIAM J. Numer. Anal.* 35, 2 (1998), 655–676. doi:10.1137/S003614299630587X

[66] J. Hoffman and S. Frankel. 2001. *Numerical Methods for Engineers and Scientists* (2nd ed.). CRC Press. doi:10.1201/9781315274508

[67] R. Hogg, J. McKean, and A. Craig. 2018. *Introduction to Mathematical Statistics* (8th ed.). Pearson, Boston.

[68] S.-M. Hu, G.-Z. Wang, and T.-G. Jin. 1996. Properties of two types of generalized ball curves. *Computer-Aided Design* 28, 2 (1996), 125–133. doi:10.1016/0010-4485(95)00047-X

[69] M. James. 1978. The Generalised Inverse. *The Mathematical Gazette* 62, 420 (1978), 109–114. doi:10.2307/3617665

[70] I. Juhász and Ṙóth. 2010. Closed rational trigonometric curves and surfaces. *J. Comput. Appl. Math.* 234, 8 (2010). doi:10.1016/j.cam.2010.03.009

[71] S. Kim and H. Moon. 2019. Gauss–Lobatto polygon of Pythagorean hodograph curves. *Computer Aided Geometric Design* 74 (2019), 101768. doi:10.1016/j.cagd.2019.101768

[72] E. Lee. 1987. The rational Bézier representation for conics. *Geometric Modeling: Algorithms and new trends* (1987), 3–19.

[73] E. Lee. 1989. Choosing nodes in parametric curve interpolation. *Computer-Aided Design* 21, 6 (1989), 363–370. doi:10.1016/0010-4485(89)90003-1

[74] S. Marchi and C. Bandiziol. 2019. On the Lebesgue constant of the trigonometric Floater–Hormann rational interpolant at equally spaced nodes. *Dolomites Research Notes on Approximation* 12, 1 (2019), 51–67. doi:10.14658/PUPJ-DRNA-2019-1-6

[75] A. Marco and J.-J. Martínez. 2007. A fast and accurate algorithm for solving Bernstein–Vandermonde linear systems. *Linear Algebra Appl.* 422, 2–3 (April 2007), 616–628. doi:10.1016/j.laa.2006.11.020

[76] R. McNeel and Associates. 2025. Rhino 7 help: Conic. https://docs.mcneel.com/rhino/7/help/en-us/commands/conic.htm Accessed: 2025-03-11.

[77] H. Moon, S. Kim, and S.-H. Kwon. 2023. Gauss–Legendre polynomial basis for the shape control of polynomial curves. *Appl. Math. Comput.* 451 (Aug. 2023), 127995. doi:10.1016/j.amc.2023.127995

[78] G. Morin and R. Goldman. 2001. On the smooth convergence of subdivision and degree elevation for Bézier curves. *Computer Aided Geometric Design* 18, 7 (2001), 657–666. doi:10.1016/S0167-8396(01)00059-0 Pierre Bézier.

[79] Y. Nakatsukasa, O. Sète, and L. Trefethen. 2018. The AAA algorithm for rational approximation. *SIAM Journal on Scientific Computing* 40, 3 (2018), A1494–A1522. doi:10.1137/16M1106122

[80] E. Neville. 1934. Iterative Interpolation. *Journal of the Indian Mathematical Society* 20 (1934), 87–120.

[81] P. Olver. 2006. On Multivariate Interpolation. *Studies in Applied Mathematics* 116, 2 (2006), 201–240. doi:10.1111/j.1467-9590.2006.00335.x

[82] R. Patterson. 1985. Projective transformations of the parameter of a Bernstein–Bézier curve. *ACM Transactions on Graphics* 4, 4 (Oct. 1985), 276–290. doi:10.1145/6116.6119

[83] H. Phien and N. Dejdumrong. 2000. Efficient algorithms for Bézier curves. *Computer Aided Geometric Design* 17, 3 (2000), 247–250. doi:10.1016/S0167-8396(99)00048-5

[84] L. Piegl. 1986. A geometric investigation of the rational Bézier scheme of computer aided design. *Computers in Industry* 7, 5 (1986). doi:10.1016/0166-3615(86)90088-6

[85] L. Piegl. 1987. On the use of infinite control points in CAGD. *Computer Aided Geometric Design* 4, 1–2 (July 1987), 155–166. doi:10.1016/0167-8396(87)90032-X

[86] L. Piegl. 1989. Modifying the shape of rational B-splines. Part 1: curves. *Computer-Aided Design* 21, 8 (1989), 509–518. doi:10.1016/0010-4485(89)90059-6

[87] L. Piegl and W. Tiller. 1997. *The NURBS Book* (2 ed.). Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-59223-2

[88] M. Planitz. 1979. Inconsistent Systems of Linear Equations. *The Mathematical Gazette* 63, 425 (1979), 181–185. doi:10.2307/3617890

[89] M. Powell. 1981. *Approximation Theory and Methods*. Cambridge University Press, Cambridge. doi:10.1017/CBO9781139171502

[90] H. Prautzsch, W. Boehm, and M. Paluszny. 2002. *Bézier and B-Spline Techniques*. Springer Berlin Heidelberg.

[91] H. Prautzsch and L. Kobbelt. 1994. Convergence of subdivision and degree elevation. *Advances in Computational Mathematics* 2, 1 (1994), 143–154. doi:10.1007/BF02519040

[92] A. Ramanantoanina and C. Fuda. 2024. Bézier [Source code]. https://github.com/mahenina0403/bezier

[93] A. Ramanantoanina and K. Hormann. 2021. New shape control tools for rational Bézier curve design. *Computer Aided Geometric Design* 88 (2021), 102003. doi:10.1016/j.cagd.2021.102003

[94] A. Ramanantoanina and K. Hormann. 2023. Shape control tools for periodic Bézier curves. *Computer Aided Geometric Design* 103 (June 2023), Article 102193, 12 pages. doi:10.1016/j.cagd.2023.102193

[95] A. Ramanantoanina and K. Hormann. 2024. Trigonometric Tangent Interpolating Curves. In *Pacific Graphics Conference Papers and Posters*, Renjie Chen, Tobias Ritschel, and Emily Whiting (Eds.). The Eurographics Association. doi:10.2312/pg.20241297

[96] M. Randrianarivony. 2011. Arc Length of Rational Bézier Curves and Use for CAD Reparametrization. *World Academy of Science, Engineering and Technology* 5, 8 (2011), 1524–1529.

[97] Ṙóth, I. Juhász, J. Schicho, and M. Hoffmann. 2009. A cyclic basis for closed curve and surface modeling. *Computer Aided Geometric Design* 26, 5 (June 2009), 528–546. doi:10.1016/j.cagd.2009.02.002

[98] C. Runge. 1901. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. *Zeitschrift für Mathematik und Physik* 46 (1901), 224–243.

[99] H. Rutishauser. 1976. *Vorlesungen über Numerische Mathematik, Band 1: Gleichungssysteme, Interpolation und Approximation*. Birkhäuser Basel. doi:10.1007/978-3-0348-5509-9

[100] H. Salzer. 1948. Coefficients for facilitating trigonometric interpolation. *Journal of Mathematics and Physics* 27, 1-4 (April 1948), 274–278. doi:10.1002/sapm1948271274

[101] H. Salzer. 1960. New formulas for trigonometric interpolation. *Journal of Mathematics and Physics* 39, 1-4 (1960), 83–96. doi:10.1002/sapm196039183

[102] H. Salzer. 1972. Lagrangian interpolation at the Chebyshev points $x_{n,\nu} \equiv \cos(\nu\pi/n)$, $\nu = 0(1)n$; some unnoted advantages. *Comput. J.* 15, 2 (05 1972), 156–159. doi:10.1093/comjnl/15.2.156

[103] J. Sánchez-Reyes. 1997. Higher order Bézier circles. *Computer-Aided Design* 29, 6 (1997), 469–472. doi:10.1016/S0010-4485(96)00084-X

[104] J. Sánchez-Reyes. 1998. Harmonic rational Bézier curves, p-Bézier curves and trigonometric polynomials. *Computer Aided Geometric Design* 15, 9 (1998), 909–923. doi:10.1016/S0167-8396(98)00031-4

[105] J. Sánchez-Reyes. 2009. Periodic Bézier curves. *Computer Aided Geometric Design* 26, 9 (Dec. 2009), 989–1005. doi:10.1016/j.cagd.2009.08.002

[106] J. Sánchez-Reyes. 2023. Shape factors and shoulder points for shape control of rational Bézier curves. *Computer-Aided Design* 157 (2023), 103477. doi:10.1016/j.cad.2023.103477

[107] K. Saniee. 2008. A Simple Expression for Multivariate Lagrange Interpolation. *SIAM Undergraduate Research Online (SIURO)* 1 (2008). Issue 1. doi:10.1137/08S010025

[108] C. Schneider and W. Werner. 1986. Some new aspects of rational interpolation. *Math. Comp.* 47, 175 (July 1986), 285–299. doi:10.2307/2008095

[109] C. Schulz. 2009. Bézier clipping is quadratically convergent. *Computer Aided Geometric Design* 26, 1 (2009), 61–74. doi:10.1016/j.cagd.2007.12.006

[110] L. Schumaker and W. Volk. 1986. Efficient evaluation of multivariate polynomials. *Computer Aided Geometric Design* 3, 2 (1986), 149–154. doi:10.1016/0167-8396(86)90018-X

[111] H. Schwarz and N. Köckler. 2011. *Numerische Mathematik*. Vieweg+Teubner Verlag Wiesbaden. doi:10.1007/978-3-8348-8166-3

[112] T. Sederberg and T. Nishita. 1990. Curve intersection using Bézier clipping. *Computer-Aided Design* 22, 9 (1990), 538–549. doi:10.1016/0010-4485(90)90039-F

[113] V. Serov. 2017. *Fourier Series, Fourier Transform and Their Applications to Mathematical Physics*. Applied Mathematical Sciences, Vol. 197. Springer International Publishing, Oulu, Finland. doi:10.1007/978-3-319-65262-7

[114] H. Speleers, P. Dierckx, and S. Vandewalle. 2007. Weight control for modelling with NURPS surfaces. *Computer Aided Geometric Design* 24, 3 (2007), 179–186. doi:10.1016/j.cagd.2007.01.005

[115] Bentley Systems. 2025. Conic curves. https://docs.bentley.com/LiveContent/web/ProStructures%20Help-v7/en/CurvesPlaceConic.html Accessed: 2025-03-11.

[116] Dassault Systèmes. 2022. SOLIDWORKS Help: Conics. https://help.solidworks.com/2022/english/solidworks/sldworks/c_conics.htm Accessed: 2025-03-11.

[117] M. Taylor, B. Wingate, and R. Vincent. 2001. An Algorithm for Computing Fekete Points in the Triangle. *SIAM J. Numer. Anal.* 38, 5 (2001), 1707–1720.

[118] W. Taylor. 1945. Method of Lagrangian curvilinear interpolation. *J. Res. Nat. Bur. Standards* 35 (Aug. 1945), 151–155. doi:10.6028/jres.035.006

[119] J. Walker. 1996. *Fast Fourier Transforms* (2nd ed.). CRC Press. doi:10.1201/9780203756188

[120] G. Wang. 1987. Ball curve of high degree and its geometric properties. *Applied Mathematics: A Journal of Chinese Universities* 2, 1 (1987), 126–140.

[121] G.-J. Wang and G.-Z. Wang. 1992. The rational cubic Bézier representation of conics. *Computer Aided Geometric Design* 9, 6 (1992), 447–455. doi:10.1016/0167-8396(92)90043-O

[122] T. Warburton. 2006. An explicit construction of interpolation nodes on the simplex. *Journal of Engineering Mathematics* 56 (11 2006), 247–262. doi:10.1007/s10665-006-9086-6

[123] J. Warren. 1993. An Efficient Algorithm for Evaluating Polynomials in the Pòlya Basis. In *Geometric Modelling, Dagstuhl, Germany, 1993 (Computing Supplementa, Vol. 10)*, Hans Hagen, Gerald E. Farin, Hartmut Noltemeier, and Rudolf F. Albrecht (Eds.). Springer, 357–361.

[124] H. Wolters. 2002. Chapter 5 - Rational Techniques. In *Handbook of Computer Aided Geometric Design*, Gerald Farin, Josef Hoschek, and Myung-Soo Kim (Eds.). North-Holland, Amsterdam, 111–140. doi:10.1016/B978-044451104-1/50006-X

[125] P. Woźny and F. Chudy. 2020. Linear-time geometric algorithm for evaluating Bézier curves. *Computer-Aided Design* 118 (2020), 102760. doi:10.1016/j.cad.2019.102760

[126] Z. Yan, S. Schiller, and S. Schaefer. 2019. Circle reproduction with interpolatory curves at local maximal curvature points. *Computer Aided Geometric Design* 72 (2019), 98–110. doi:10.1016/j.cagd.2019.06.002

[127] Z. Yan, S. Schiller, G. Wilensky, N. Carr, and S. Schaefer. 2017-07. κ-Curves: Interpolation at Local Maximum Curvature. *ACM Transactions on Graphics* 36, 4 (2017-07). doi:10.1145/3072959.3073692

[128] B. Ycart. 2013. A case of mathematical eponymy: the Vandermonde determinant. *Revue d'histoire des mathématiques* 19, 1 (2013), 43–77. doi:10.24033/rhm.168

[129] C. Yuksel. 2020-08. A class of C^2 interpolating splines. *ACM Transactions on Graphics* 39, 5 (2020-08). doi:10.1145/3400301

[130] C. Zhang, F. Cheng, and K. Miura. 1998. A method for determining knots in parametric curve interpolation. *Computer Aided Geometric Design* 15, 4 (1998), 399–416. doi:10.1016/S0167-8396(97)00041-1

[131] D. Zwillinger. 2003. *CRC Standard Mathematical Tables and Formulae*. CRC Press.