Trigonometric Tangent Interpolating Curves

A. Ramanantoanina[1] and K. Hormann[1]

[1]Università della Svizzera italiana, Lugano, Switzerland

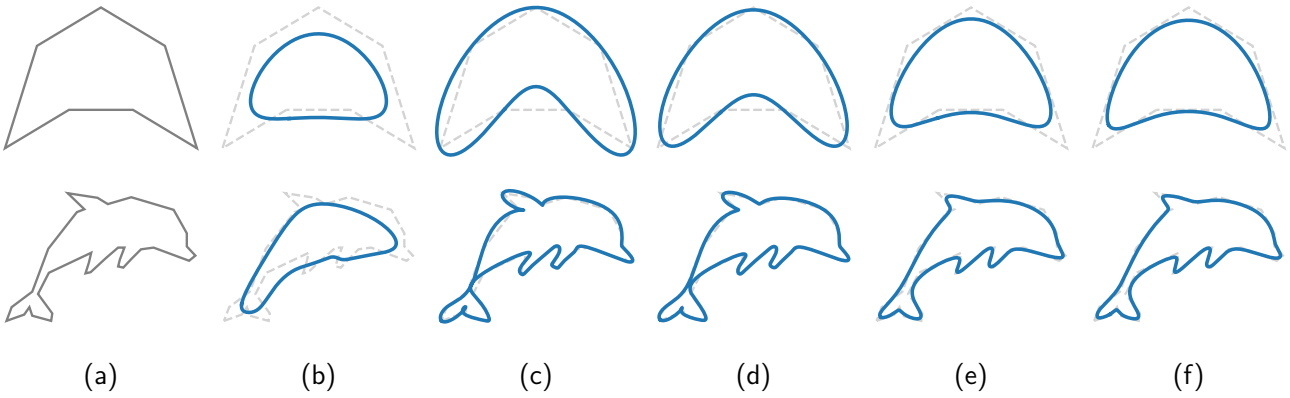

Figure 1: *Comparison of curves defined by a common closed control polygon (a): while periodic Bézier curves suffer from shrinking as the number of control points increases (b), trigonometric vertex interpolating curves tend to have shape artefacts in the presence of control edges with non-uniform length. 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).*

Abstract

Due to their favourable properties, cubic B-spline curves are the de facto standard for modelling closed curves in computer graphics and computer-aided design. Their shapes can be modified intuitively by moving the vertices of a control polygon, but they are only twice differentiable at the knots. Even though this is sufficient for most applications, curves with higher smoothness are still of valuable interest. For example, periodic Bézier curves provide an alternative for designing closed curves as C^∞ smooth trigonometric polynomials, but their shapes are not as intuitive to control, because of the global influence of each control point. The same space of curves can also be described in vertex interpolating form, but this may result in other shape artefacts. In this paper we introduce two new representations of trigonometric polynomial curves that are inspired by the idea behind polynomial Gauss–Legendre curves and likewise use the control polygon for controlling the tangents of the curves. The first variant gives curves that closely follow the control polygon, and the curves generated with the second variant are less tied to the control polygon and instead very similar to uniform cubic B-spline curves.

CCS Concepts

• ***Computing methodologies*** → ***Parametric curve and surface models;***

1. Introduction

A *trigonometric polynomial of degree N* is a function

$$P(t) = a_0 + \sum_{k=1}^{N} \big(a_k \cos(kt) + b_k \sin(kt)\big)$$

with *coefficients* $a_0, \dots, a_N, b_1, \dots, b_N$. Clearly, P is C^∞ smooth and periodic in t over $[0, 2\pi]$. In this paper we focus on the case where the coefficients are points in $\mathbb{R}^2$, so that $P(t)$ for $t \in [0, 2\pi]$ describes a closed curve. For example, if $N = 1$, then $P(t)$ is an ellipse, and the coefficients a_0, a_1, b_1 encode its centre and the direction of its principal axes. However, for a complex shape, this formulation does not provide much geometric information about the curve, and it is preferable to express $P(t)$ in a different basis.

DOI: 10.2312/pg.20241297

One option is to let $n = 2N$ and to consider as basis functions the $n+1$ uniformly shifted versions of

$$B_0^n(t) = \frac{2^n}{n+1}\binom{n}{N}^{-1}\cos^n\frac{t}{2},$$

that is,

$$B_i^n(t) = B_0^n(t-\phi_i), \qquad i = 0,\dots,n, \tag{1}$$

where $\phi_i = 2i\pi/(n+1)$, and to write $P(t)$ as

$$P(t) = \sum_{i=0}^{n} B_i^n(t)\,p_i$$

for certain *control points* $p_0,\dots,p_n \in \mathbb{R}^2$. Curves given in this form are called *periodic Bézier curves* [RJSH09, SR09], since they share some key properties with polynomial Bézier curves. 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 functions B_i^n in (1) are designed to be somewhat similar to the Bernstein polynomials, which are used for expressing 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 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 1b).

Another option is to use Gauss's formula for trigonometric interpolation [Gau66] and express $P(t)$ in interpolating form as

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

where

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

are the *trigonometric Lagrange basis functions* with respect to the nodes $t_0,\dots,t_n$ [Sal48]. Like in the polynomial case, the functions ℓ_i^n satisfy the Lagrange property $\ell_i(t_j) = \delta_{i,j}$, which guarantees the interpolation property, namely $P(t_i) = p_i$ (see Figure 1c). In the special case of equidistant nodes, $t_i = \phi_i$, one can further express the curve in trigonometric barycentric form [Ber84], which allows for a very efficient evaluation of $P(t)$. However, due to the oscillatory nature of this basis (see Figure 3), this representation is suitable only for small displacements of the control points p_i [RH23].

1.1. Contributions

In this paper, we explore two novel alternative representations of trigonometric curves, both based on the interpolation of tangents instead of points (Section 3). The first construction gives curves that are very tight to the control polygon (see Figure 1d). The second construction is a minor variation of the first, but turns out to give curves (see Figure 1e) that are very similar to cubic B-spline curves (see Figure 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.

After deriving our trigonometric tangent interpolating curves and reporting their basic properties (Section 3), we discuss some practical aspects of using them for curve modelling (Section 4) and point out advantages and limitations (Section 5).

2. Preliminaries

Before starting the construction, let us recall some trigonometric identities that we use frequently throughout this manuscript:

$$\cos(\alpha-\beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta, \tag{3}$$

$$\sin\alpha - \sin\beta = 2\cos\frac{\alpha+\beta}{2}\sin\frac{\alpha-\beta}{2}, \tag{4}$$

$$\cos^{2m}t = \frac{1}{2^{2m}}\binom{2m}{m} + \frac{1}{2^{2m-1}}\sum_{i=0}^{m-1}\binom{2m}{i}\cos\big(2(m-i)t\big). \tag{5}$$

We further introduce the vector

$$\boldsymbol{c}_n(t) = \big(1,\cos t,\sin t,\dots,\cos(Nt),\sin(Nt)\big)^T \in \mathbb{R}^{n+1},$$

of trigonometric monomials and the matrix

$$C_{n,m} = \big(\boldsymbol{c}_n(\phi_0),\dots,\boldsymbol{c}_n(\phi_m)\big)^T \in \mathbb{R}^{(m+1)\times(n+1)}.$$

3. Trigonometric tangent interpolating curves

Suppose 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 [MKK23]. 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 2, top), that is,

$$P_1'(\psi_i) = \omega_i\Delta_i, \qquad i = 0,\dots,n, \tag{6}$$

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 (6) are associated with the edge midpoints of the control polygon, we further propose a variant where the constraints are associated with the control points instead. To this end, we simply average two 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 2, bottom), that is,

$$P_2'(\phi_i) = \omega_i\frac{\Delta_{i-1}+\Delta_i}{2}, \qquad i = 0,\dots,n, \tag{7}$$

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

In order to derive the appropriate basis functions L_i^d that allow

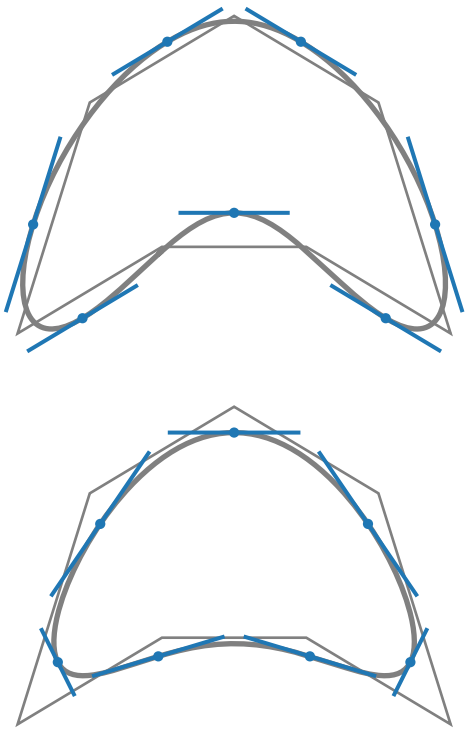

Figure 2: *Tangent interpolation conditions (9) for $n = 6$ at the highlighted points $P_d(t_i)$ for type-1 curves (top) and type-2 curves (bottom).*

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

$$P_d(t) = \sum_{i=0}^{n} L_i^d(t)\, p_i, \tag{8}$$

note that we can write the interpolation conditions in (6) and (7) conveniently in a common form as

$$P_d'(t_i) = \omega_i \frac{p_{i+1} - p_{i+1-d}}{d}, \qquad i = 0, \dots, n, \tag{9}$$

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

For the rest of the manuscript, 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 (9) and the basis functions in (2), as

$$P_d'(t) = \sum_{i=0}^{n} \ell_i^n(t)\, \omega_i \frac{p_{i+1} - p_{i+1-d}}{d}. \tag{10}$$

Integrating both sides of (10), we get P_d as in (8) with basis functions

$$L_i^d(t) = \frac{1}{d}\big(\omega_{i-1} I_{i-1}(t) - \omega_{i-1+d} I_{i-1+d}(t)\big) \tag{11}$$

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

$$I_i(t) = \int_{t_i}^{t} \ell_i^n(x)\, \mathrm{d}x \tag{12}$$

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 get 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 (2) becomes

$$\ell_i^n(t) = K(t - t_i), \tag{13}$$

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 [Pow81]. We can now use (13) to write $I_i(t)$ in (12) as

$$I_i(t) = \int_0^{t - t_i} K(x)\, \mathrm{d}x = \frac{1}{n+1}\left(t - t_i + \sum_{k=1}^{N} \frac{2}{k} \sin\big(k(t - t_i)\big)\right), \tag{14}$$

which in turn can be used in (11) to get an explicit formula for L_i^d.

3.1. 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 (11) and (14) that

$$\begin{aligned} L_i^d(t) &= \frac{\omega}{d}\big(I_{i-1}(t) - I_{i-1+d}(t)\big) \\ &= \frac{\omega}{d(n+1)}\left(\phi_d + \sum_{k=1}^{N} \frac{2}{k}\Big[\sin\big(k(t - t_{i-1})\big) \right. \\ &\qquad \left. - \sin\big(k(t - t_{i-1+d})\big)\Big]\right) \end{aligned} \tag{15}$$

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 (15) and using (4), we can finally simplify the formula for L_i^d to

$$L_i^d(t) = \frac{1}{n+1} + \frac{2}{d\pi} \sum_{k=1}^{N} \frac{1}{k} \cos\big(k(t - \phi_i)\big) \sin\frac{k\phi_d}{2}. \tag{16}$$

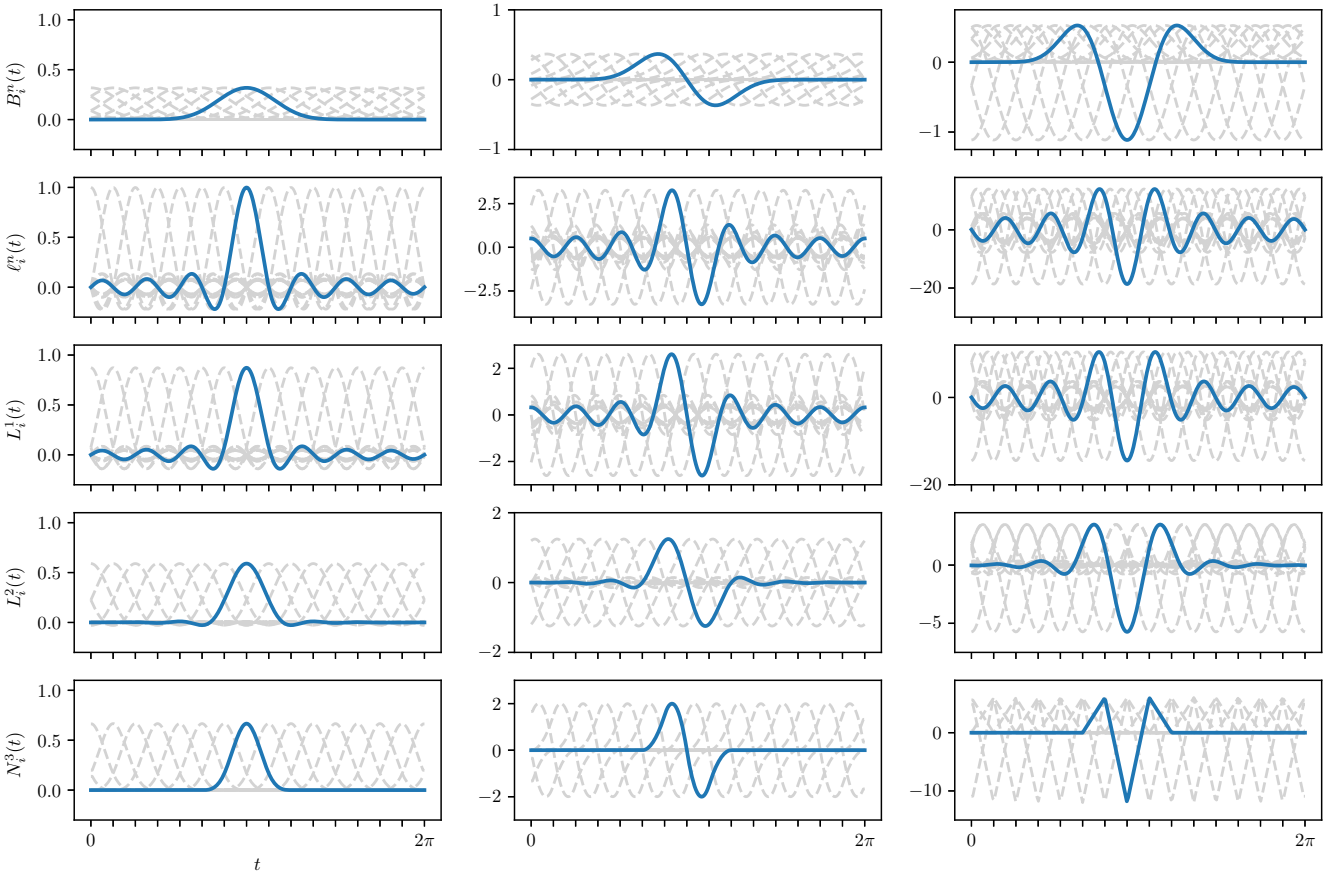

Figure 3: *A comparison of the different basis functions considered in this paper (left column) for $n = 14$ and the corresponding first derivatives (middle column) and second derivatives (right column). From top to bottom: periodic Bézier, trigonometric Lagrange, trigonometric tangent interpolating type-1 and type-2, and uniform cubic B-splines.*

3.2. 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 $\boldsymbol{L}_n^d(t) = \left(L_0^d(t), \dots, L_n^d(t)\right)^T$ can be expressed as $\boldsymbol{L}_n^d(t) = \Gamma_{n,d}\boldsymbol{c}_n(t)$ for some non-singular matrix $\Gamma_{n,d}$.

To this end, we use (3) to expand $L_i^d(t)$ in (16) 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

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

where D_d is the diagonal matrix

$$D_d = \operatorname{diag}(a_0, a_1, a_1, \dots, a_N, a_N),$$

with entries

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

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

$$\Gamma_{n,d} = C_{n,n}D_d,$$

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 $i = 1, \dots, N$. On the other hand, we recall that the functions $1, \cos t, \sin t, \dots, \cos(Nt), \sin(Nt)$ form a Chebyshev system, that is, they are linearly independent and no non-trivial linear combination of them admits more than n zeros in $[0, 2\pi)$ [Pow81]. Therefore, D_d and $C_{n,n}$ are both non-singular.

4. Practical aspects

4.1. Implementation

The implementation of our tangent interpolating curves is straightforward. Given some parameter $t \in [0, 2\pi]$, we first evaluate each of

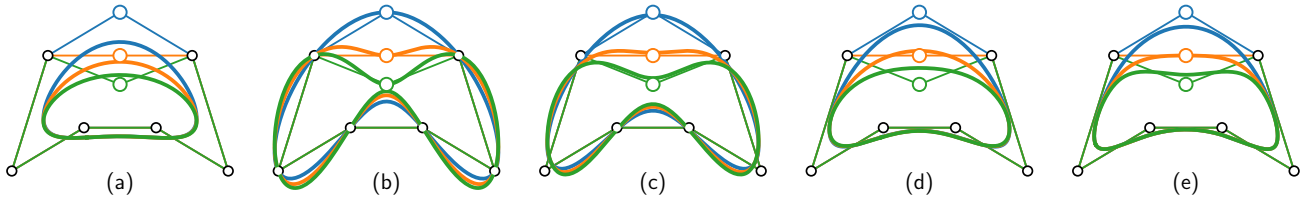

Figure 4: *Effect of modifying a single control point for various curve types: periodic Bézier curve (a), trigonometric vertex interpolating curve (b), trigonometric tangent interpolating curve of type 1 (c) and type 2 (d), and cubic B-spline curve (e).*

the $n+1$ basis functions L_i^d in $O(n)$ time, using (16), and then compute the corresponding curve point $P_d(t)$ using (8) 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. 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.

4.2. Curve manipulation

Since the basis functions of our type-1 and type-2 curves are neither non-negative nor non-positive (see Figure 3), that is, they do not form a blending system [Car99], the curves are not necessarily contained in the convex hull of the control points (see Figure 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 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 5). However, the cusp point does not coincide with the overlapping control points, as it would in the case of a cubic B-spline.

4.3. Curve similarities

The examples in Figures 1, 4, and 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 3, this is not surprising, 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

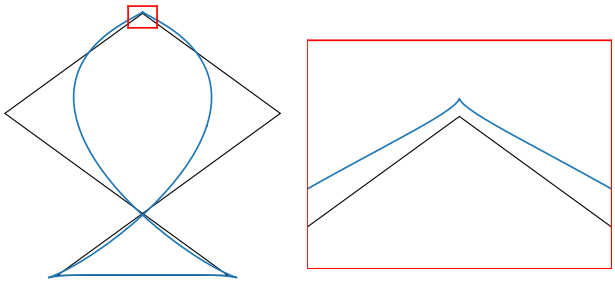

Figure 5: *An example of a curve given by 11 points with sharp corners (top, bottom left, and bottom right) created by overlapping three successive points with a zoomed-in view of the top corner.*

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 $j \neq i-1, i+1$.

4.4. 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 $(p_0, \ldots, p_n)$ and $(q_0, \ldots, q_m)$, respectively. Since $\boldsymbol{L}_i^d(t) = \Gamma_{i,d}\boldsymbol{c}_i(t)$ for $i = n, m$, we can get 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

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

can be precomputed.

Since we are in a cyclic setting and more focused on the shape rather than the parameterization, we can shift the indices of the

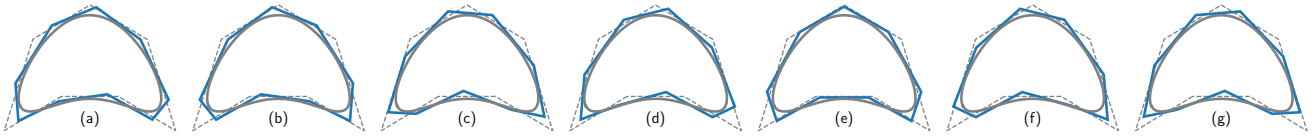

Figure 6: *Set of all degree-elevated control polygons* $\boldsymbol{q}$ *(blue) for different choices of* p_0*. In this example, the optimization (17) picks the control polygon (e) as the preferred degree-elevated control polygon, which is the one that keeps the reflection symmetry present in* $\boldsymbol{p}$ *(dashed).*

control points of $\boldsymbol{p}$ arbitrarily 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 by simply computing all of them and picking the one that minimizes 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 [HMC19] that the variance of the edge lengths of $\boldsymbol{p}$ is defined as

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

where

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

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

$$|\sigma^2(\boldsymbol{p}) - \sigma^2(\boldsymbol{q})|. \tag{17}$$

Figure 6 shows an example of this procedure.

4.5. Basis transformations

Each method for designing trigonometric curves (see Figure 4) has their particular advantages. While 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 know as basis transformation (see Figure 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 $p_0,\dots,p_n$ as in (8), we would like to find the control points $q_0,\dots,q_n$, such that same curve can be expressed in Bézier form as

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

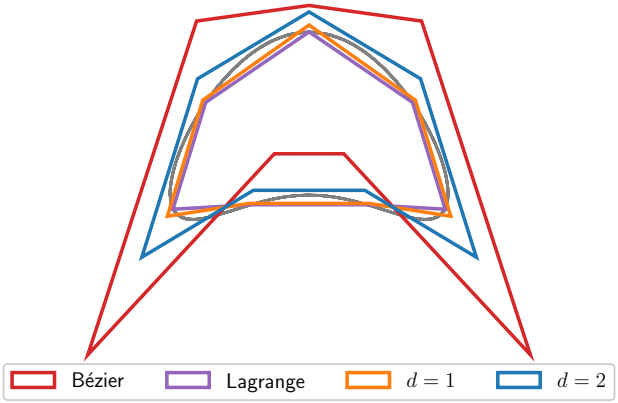

Figure 7: *Control polygons of the same trigonometric polynomial curve for different sets of basis functions.*

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

$$\begin{aligned}
B_i^n(t) &= \frac{2^n}{n+1}\binom{n}{N}^{-1}\cos^n\frac{t-\phi_i}{2}\\
&\overset{(5)}{=} \frac{1}{n+1}\binom{n}{N}^{-1}\left[\binom{n}{N} + 2\sum_{k=0}^{N-1}\binom{n}{k}\cos[(N-k)(t-\phi_i)]\right]\\
&= \frac{1}{n+1}\binom{n}{N}^{-1}\left[\binom{n}{N} + 2\sum_{k=1}^{N}\binom{n}{N-k}\cos[k(t-\phi_i)]\right]\\
&\overset{(3)}{=} \frac{1}{n+1} + \frac{2}{n+1}\binom{n}{N}^{-1}\sum_{k=1}^{N}\binom{n}{N-k}\cos(kt)\cos(k\phi_i)\\
&\qquad + \frac{2}{n+1}\binom{n}{N}^{-1}\sum_{k=1}^{N}\binom{n}{N-k}\sin(kt)\sin(k\phi_i),
\end{aligned}$$

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

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

Letting $\boldsymbol{B}_n(t)$ denote the vector $\boldsymbol{B}_n(t) = \big(B_0^n(t),\dots,B_n^n(t)\big)^T$, we

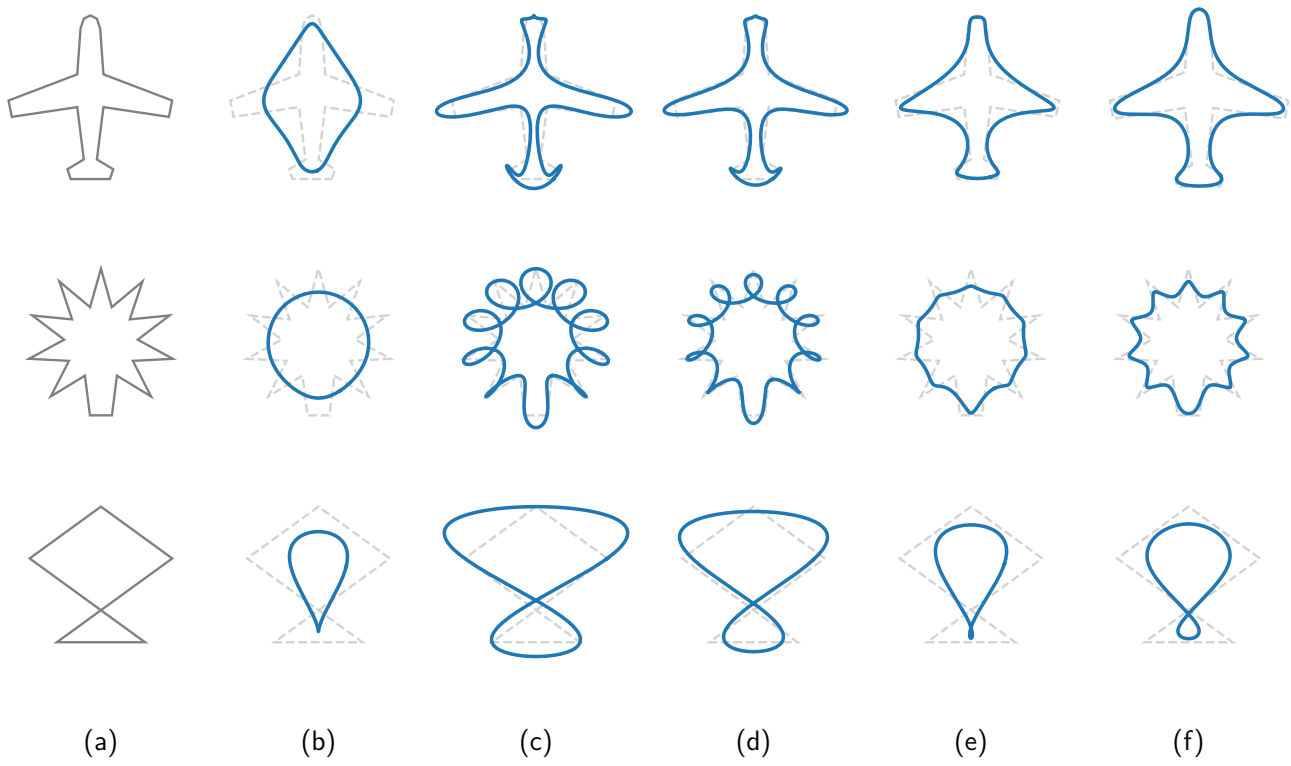

Figure 8: *Another comparison of periodic Bézier curves (b), vertex interpolating curves (c), our tangent interpolating type-1 (d) and type-2 curves (e), and cubic B-spline curves (f), all defined by the same control polygon (a); cf. Figure 1. This example highlights the result given by control polygons with a greatly varying control edge lengths (top), a zig-zag pattern (middle) and a self-intersection (bottom).*

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 $(p_0, \dots, p_n)$ and $(q_0, \dots, q_n)$, 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 swap between the control polygons $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ of 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 [RH23].

5. Conclusion

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 offer 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.

However, one limitation of trigonometric polynomial curves is that they are restricted to an odd number of control points. While this is not a severe constraint, it would be interesting to study how control polygons with an even number of control points can fit into the picture, since this may sometimes be the natural choice for modelling curves with certain symmetric features (e.g., an even degree rotational symmetry).

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 ratio-

nal Bézier curves [RH23] can be expressed in tangent interpolating form with additional shape parameters.

Finally, it is also well known from B-spline curves that uniform nodes are not the proper choice in case of control polygons whose 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 non-uniform nodes.

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions, which helped to improve this paper. This work was supported by the Swiss National Science Foundation (SNF) under project No. 188577 and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 860843.

References

[Ber84] BERRUT J.-P.: Baryzentrische Formeln zur trigonometrischen Interpolation (I). *Zeitschrift für angewandte Mathematik und Physik 35*, 1 (Jan. 1984), 91–105. doi:10.1007/BF00945179. 2

[Car99] CARNICER J. M.: Interpolation, shape control and shape properties. In *Shape Preserving Representations in Computer-Aided Geometric Design*, Peña J. M., (Ed.). Nova Science Publishers, Commack, 1999, ch. 2, pp. 15–43. 5

[Gau66] GAUSS C. F.: Theoria interpolationis methodo nova tractata. In *Werke*, vol. 3. Königliche Gesellschaft der Wissenschaften, Göttingen, 1866, pp. 265–330. 2

[HMC19] HOGG R. V., MCKEAN J. W., CRAIG A. T.: *Introduction to Mathematical Statistics*, 8th ed. Pearson, Boston, 2019. 6

[MKK23] MOON H. P., KIM S. H., KWON S.-H.: Gauss–Legendre polynomial basis for the shape control of polynomial curves. *Applied Mathematics and Computation 451* (Aug. 2023), Article 127995, 16 pages. doi:10.1016/j.amc.2023.127995. 2

[Pow81] POWELL M. J. D.: *Approximation Theory and Methods*. Cambridge University Press, Cambridge, 1981. doi:10.1017/CBO9781139171502. 3, 4

[RH23] RAMANANTOANINA A., HORMANN K.: 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. 2, 7, 8

[RJSH09] RÓTH Á., JUHÁSZ I., SCHICHO J., HOFFMANN M.: 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. 2

[Sal48] SALZER H. E.: Coefficients for facilitating trigonometric interpolation. *Journal of Mathematics and Physics 27*, 1–4 (Apr. 1948), 274–278. doi:10.1002/sapm1948271274. 2

[SR09] SÁNCHEZ-REYES J.: Periodic Bézier curves. *Computer Aided Geometric Design 26*, 9 (Dec. 2009), 989–1005. doi:10.1016/j.cagd.2009.08.002. 2