www.cad-journal.net
Computer-Aided Design AND APPLICATIONS™

Beta-B-Spline Curves

Seifalla A. Moustafa[1], Fuhua (Frank) Cheng[1], Shuhua Lai[2], Alice J. Lin[3], Anastasia Kazadi[4]

[1]University of Kentucky, author@cs.uky.edu
[2]Georgia Gwinnett College, author@ggc.edu
[3]Austin Peay State University, author@apsu.edu
[4]Eastern Kentucky University, author@eku.edu

Corresponding author: Seifalla A. Moustafa, seifalla.moustafa@uky.edu

Abstract. This paper presents β-B-spline curves, which extend the classical B-spline formulation by introducing a tension parameter β derived from the properties of the beta function. By adapting the de Boor algorithm, we demonstrate that curve shapes can be modified through β without altering their control points, maintaining the usual geometric properties while providing an additional and intuitive means of control.

Keywords: B-Splines, Bezier Curve, De Boor Algorithm
DOI: https://doi.org/10.14733/cadaps.2027.aaa-bbb

1 INTRODUCTION

For most of the applications in computer-aided design (CAD), computer graphics, image processing and computer vision, people use parametric or implicit curves/surfaces to represent 2D/3D shapes. Parametric curves/surfaces such as composite Bezier curves/surfaces, B-spline curves/surfaces, NURBS curves/surfaces or subdivision surfaces are preferred in the CAD area because one can adjust or fine tune the shape of a curve/surface by adjusting its control points locally.

However, for complicated shapes (which is the case in most CAD applications), adjusting the shape of a curve/surface by interactively adjusting related control points, even just locally, is a laborious and time-consuming job. So people were wondering if there were other ways or more flexible ways for us to adjust the shape of a curve or surface instead of just modifying its control points.

For years people tried to find ways to extend/modify the definition of Bézier and B-spline curves so that one could change the shape of the curve without changing the control points of the curve [3, 4, 6, 7, 8]. But none of the works seem to be intuitive enough for practical applications in the field.

Adding a tension parameter in the representation of a curve or surface seems to be a good option in that direction. One can adjust the shape of a curve or surface simply by adjusting the value of the tension parameter globally or locally.

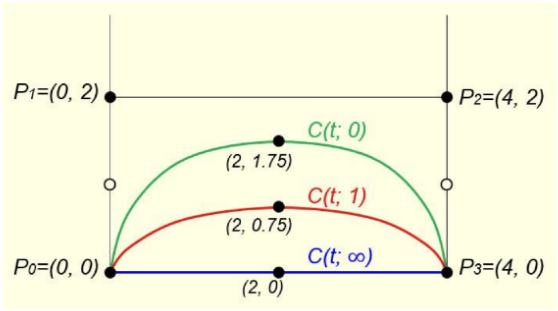


Figure 1: Shape Variation in the Bézier Curve Case

Being able to adjust the shape of a curve or surface locally is especially important for applications that emphasize visual effects through a morphing process such as cartoon/game figure design, surgery simulation, and TV special effects.

In this paper, we propose β-B-spline curves, which extend the classical B-spline formulation by introducing a tension parameter β derived from properties of the beta function. By adapting the de Boor algorithms, we demonstrate that curve shapes can be modified through β without altering their control points (see figure 1), maintaining the usual geometric properties while providing an additional and intuitive means of control.

The remainder of this paper is organized as follows: Section 2 overviews related work. Section 3 presents our proposed method. Section 4 discusses some results. Finally, section 5 summarizes our concluding remarks.

2 RELATED WORK

Many other works proposed approaches similar to ours that are based on different types of shape parameters. Yang et al. [9] devised an interesting extension to the definition of a Bézier curve which we detail below:

Given control points $\mathbf{V}_i \in \mathbf{R}^d (d = 2, 3; i = 0, 1, \ldots, n; n \geq 2)$ and $\lambda \in [-1, 1]$, we call

$$\mathbf{p}(t) = \sum_{i=0}^{n} b_{i,n}(t)\mathbf{V}_i, \quad t \in [0, 1], \tag{1}$$

a λ-Bézier curve of order n, where $b_{i,n}(t)(i = 0, 1, \ldots, n)$ are the λ-Bernstein basis functions defined as

$$\left(1 + \frac{3C_{n-2}^{i-1} + C_{n-1}^{i} - C_n^i}{C_n^i}\lambda - \frac{2C_{n-1}^i}{C_n^i}\lambda t + \lambda t^2\right) C_n^i t^i (1-t)^{n-i}, \quad i = 0, 1, \ldots, n, n \geq 2. \tag{2}$$

Figure 2 shows the effect of λ on a λ-Bézier curve. Comparing figures 1 and 2, we see that because β spans a wider range $[0, \infty)$ compared to $\lambda \in [-1, 1]$, it allows for a greater degree of shape variation.

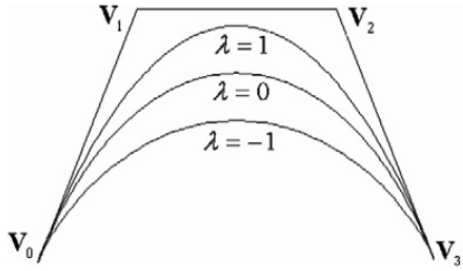


Figure 2

Another extension to the definition was presented in [4] and it is as follows: Bézier curves of degree n with n shape parameters are defined as

$$\hat{P}(t) = \sum_{i=0}^{n} \hat{B}_{i,n}(t) P_i, \quad t \in [0,1], \tag{3}$$

where P_i is a collection of control points in $\mathbb{R}^m$, $\hat{B}_{i,n}(t)$ are known as the Bernstein basis functions of degree n with n shape parameters and are defined as

$$\hat{B}_{i,n}(t) = B_{i,n}(t)\,(1 + x_{i,1}(1-t) + x_{i,2}t) \tag{4}$$

for $t \in [0,1], i = 0, 1, \ldots, n$, and $x_{i,j}(1-t)$ are defined as:

$$x_{i,1} = \frac{\lambda_i}{n-i+1}, \quad x_{i,2} = -\frac{\lambda_{i+1}}{i+1}, \quad \lambda_0 = \lambda_{n+1} = 0, \quad -(n-i+1) < \lambda_i < ii = 0, 1, \ldots, n. \tag{5}$$

Figure 3 shows the effect of the shape parameters on a Bézier curve. Although the effect of the shape parameters is predictable and can be accurately described, changing multiple parameters (either interactively or autonomously) is tedious in most cases. For a detailed analysis of the parameter behavior, the reader is referred to [4].

Kovács and Várady [5] introduced P-Bézier and P-Bspline curves as proximity-based extensions of classical Bézier and B-spline curves. Their approach introduces a proximity parameter that allows a curve to move continuously between a base curve and its control polygon, thereby providing an additional shape-control mechanism without changing the control points. This is closely related to the goal of the present work, since both approaches seek to modify the shape of a curve through a parameter rather than by manually editing the control polygon. However, their method is based on proximity control and finite-difference-based representations, whereas our construction is derived from the beta function and leads to β-Bernstein basis functions together with corresponding β-Bézier and β-B-spline evaluation algorithms.

3 PROPOSED METHOD

A famous function [2], in mathematics is the beta-function:

$$b(\alpha, \gamma) = \int_0^1 x^{\alpha-1}(1-x)^{\gamma-1}\,dx. \tag{6}$$

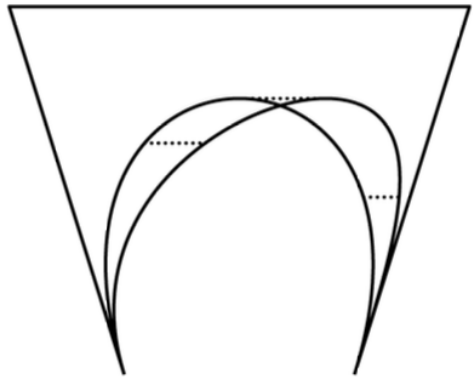


Figure 3

Originally, in this formula, β was used in place of γ. We use γ because β has a different meaning in this paper. An important property of the beta-function is the recursion property:

$$b(\alpha+1,\gamma) = \frac{\alpha}{\alpha+\gamma}\, b(\alpha,\gamma), \tag{7}$$

$$b(\alpha,\gamma+1) = \frac{\gamma}{\alpha+\gamma}\, b(\alpha,\gamma). \tag{8}$$

Let $\alpha = \lambda t$ and $\gamma = \lambda(1-t)$. Applying (7) k times and then (8) $n-k$ times, we get

$$b(\lambda t + k, \lambda(1-t)+n-k) = \frac{\prod_{i=0}^{k-1}(\lambda t+i)\ \prod_{j=0}^{n-k-1}(\lambda(1-t)+j)}{\prod_{m=0}^{n-1}(\lambda+m)}\, b(\lambda t, \lambda(1-t)). \tag{9}$$

If we multiply this by $\binom{n}{k}$ and divide it by $b(\lambda t, \lambda(1-t))$, we get a formula for a Bezier curve with a parameter in it, also known as a β-Bézier curve:

$$B_k^n(t;\lambda) = \binom{n}{k} \frac{\prod_{i=0}^{k-1}(\lambda t+i)\ \prod_{j=0}^{n-k-1}(\lambda(1-t)+j)}{\prod_{m=0}^{n-1}(\lambda+m)}. \tag{10}$$

In the limit as $\lambda \to \infty$, the formula converges to

$$\lim_{\lambda\to\infty} \binom{n}{k} \frac{\prod_{i=0}^{k-1}(\lambda t+i)\ \prod_{j=0}^{n-k-1}(\lambda(1-t)+j)}{\prod_{m=0}^{n-1}(\lambda+m)} = \binom{n}{k} t^k (1-t)^{n-k} \tag{11}$$

which is the Bernstein polynomials used in the formula for a Bezier curve. This was developed by Zeng et al. [2].

It has been shown [1] that the properties of β–Bézier curves are easier to study when $\lambda = \frac{1}{\beta}$. In this case, we have

$$B_k^n(t;\beta) = \binom{n}{k} \frac{\prod_{i=0}^{k-1}(t+i\beta)\ \prod_{j=0}^{n-k-1}((1-t)+j\beta)}{\prod_{m=0}^{n-1}(1+m\beta)}. \tag{12}$$

This is what we call the β-Bernstein basis functions which are used in the formula for a β–Bézier curve segment:

$$C(t;\beta) = \sum_{k=0}^{n} P_k \, B_k^n(t;\beta) \tag{13}$$

In this formula, P_k are the control points of the curve.

The De Casteljau algorithm is at the heart of Bézier curves. It turns out that there is a De-Casteljau-like algorithm for β–Bézier curves:

Algorithm 1 β–Bézier De Casteljau Algorithm

for $i = 0$ to n **do**
    $P_i^0 \leftarrow P_i$
end for
for $i = 1$ to n **do**
    for $j = i$ to n **do**
        $P_j^i \leftarrow \frac{1-t+(n-j)\beta}{1+(n-i)\beta} P_{j-1}^{i-1} + \frac{t+(j-i)\beta}{1+(n-i)\beta} P_j^{i-1}$
    end for
end for

We know that

$$t = \frac{x - t_k}{t_{k+1} - t_k}.$$

If we substitute this into line 6 of the β–Bézier De Casteljau algorithm, we obtain the following algorithm:

Algorithm 2 Iterative β–de Boor Algorithm

1: **for** $j = 0$ to p **do**
2: $\quad d[j] \leftarrow c[j + k - p]$
3: **end for**
4: **for** $r = 1$ to p **do**
5: $\quad$ **for** $j = p$ down to r **do**
6: $\quad\quad$ **if** $t[j + k - r + 1] - t[j + k - p] = 0$ **then**
7: $\quad\quad\quad \alpha \leftarrow 0$
8: $\quad\quad$ **else**
9: $\quad\quad\quad \alpha \leftarrow \dfrac{x - t[j + k - p]}{t[j + k - r + 1] - t[j + k - p]}$
10: $\quad\quad$ **end if**
11: $\quad\quad d[j] \leftarrow \dfrac{(1 - \alpha) + (p - j)\beta}{1 + (p - r)\beta}\, d[j - 1] + \dfrac{\alpha + (j - r)\beta}{1 + (p - r)\beta}\, d[j]$
12: $\quad$ **end for**
13: **end for**
14: **return** $d[p]$

The iterative β-de Boor algorithm has a time complexity of $O(p^2)$, where p is the spline degree. The initialization loop executes $p+1$ times, while the nested loops perform

$$\sum_{r=1}^{p}(p-r+1) = \frac{p(p+1)}{2}$$

iterations in total. Since the computations inside each iteration consist only of constant-time arithmetic and vector operations, the overall complexity remains quadratic in the spline degree. The algorithm requires $O(p)$ space due to the temporary array used to store intermediate control points.

The numerical stability of the modified de Boor algorithm depends on the behavior of the blending coefficients. Each update has the form

$$d_j^{(r)} = a\, d_{j-1}^{(r-1)} + b\, d_j^{(r-1)},$$

where

$$a = \frac{(1-\alpha)+(p-j)\beta}{1+(p-r)\beta}, \qquad b = \frac{\alpha+(j-r)\beta}{1+(p-r)\beta}.$$

Since

$$a + b = 1,$$

each update is an affine combination of two previous points. This is a favorable property for stability because the algorithm does not introduce unnecessary translations or scalings.

When $\beta = 0$, the method reduces to the standard de Boor algorithm. In that case, if $0 \le \alpha \le 1$, the coefficients form a convex combination, and the algorithm is numerically stable.

For $\beta \neq 0$, the main source of numerical instability occurs when the denominator

$$1 + (p-r)\beta$$

is close to zero. In that case, the coefficients a and b can become very large, which may amplify roundoff error. Therefore, the algorithm becomes unstable near the values

$$\beta = -\frac{1}{p-r}, \qquad r = 1, \ldots, p-1.$$

Even when the denominator is not close to zero, the method may lose the convex-combination property. If either coefficient becomes negative or larger than 1, the update is still affine, but it is no longer convex. This can cause overshoot and increased sensitivity to small perturbations in the control points.

Thus, a practical stability condition is

$$|1 + (p-r)\beta| \ge \varepsilon$$

for some positive tolerance ε, and for all

$$r = 1, \ldots, p-1.$$

A simple sufficient condition is

$$\beta \ge 0,$$

because then

$$1 + (p-r)\beta \ge 1.$$

Therefore, for nonnegative β, the denominators remain bounded away from zero, and the algorithm is expected to be reasonably stable.

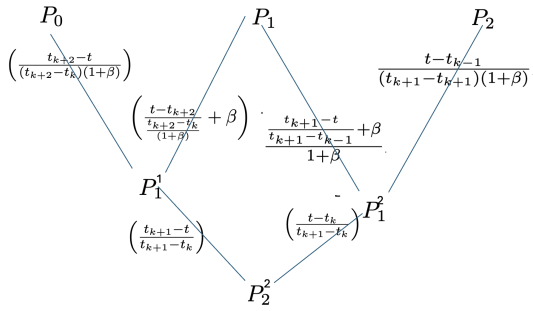
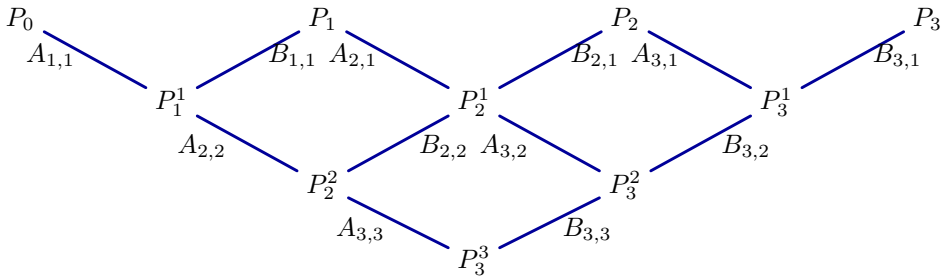


Figure 4: Pascal Triangle for the Quadratic Case

This algorithm constitutes a procedural method for generating what we call β-B-Spline curves, which are in essence B-Spline curves but with a parameter β in their formula. Figure 4 and the below figures show derivations of the formulae for a cubic and a quadratic B-Spline curves:

$$P_2^2 = \frac{t_{k+1}-t}{t_{k+1}-t_k}\left(\frac{t_{k+2}-t}{(t_{k+2}-t_k)(1+\beta)}P_0 + \frac{\frac{t-t_k}{t_{k+2}-t_k}+\beta}{1+\beta}P_1\right) + \frac{t-t_k}{t_{k+1}-t_k}\left(\frac{\frac{t_{k+1}-t}{t_{k+1}-t_{k-1}}+\beta}{1+\beta}P_1 + \frac{t-t_{k-1}}{(t_{k+1}-t_{k-1})(1+\beta)}P_2\right). \tag{14}$$



$$P_3^3 = w_0\,P_0 + w_1\,P_1 + w_2\,P_2 + w_3\,P_3, \tag{15}$$

$$w_0 = A_{3,3}\,A_{2,2}\,A_{1,1}, \tag{16}$$

$$w_1 = A_{3,3}(A_{2,2}B_{1,1} + B_{2,2}A_{2,1}) + B_{3,3}\,A_{3,2}A_{2,1}, \tag{17}$$

$$w_2 = A_{3,3}\,B_{2,2}B_{2,1} + B_{3,3}(A_{3,2}B_{2,1} + B_{3,2}A_{3,1})\,, \tag{18}$$

$$w_3 = B_{3,3}\,B_{3,2}\,B_{3,1}. \tag{19}$$

$$A_{j,r} = \frac{(1-\alpha_{j,r}) + (3-j)\beta}{1+(3-r)\beta}, \tag{20}$$

$$B_{j,r} = \frac{\alpha_{j,r} + (j-r)\beta}{1+(3-r)\beta}, \tag{21}$$

$$\alpha_{j,r} = \frac{t - t_{j+k-3}}{t_{j+k-r+1} - t_{j+k-3}} \tag{22}$$

Our extension of B-Spline curves preserves many of their classical properties, namely affine invariance, partition of unity, convex hull, and endpoint interpolation in appropriate knot configurations.

The β-de Boor curve is invariant under affine transformations. Each iteration of the algorithm updates an intermediate point using

$$d[j] \leftarrow a\, d[j-1] + b\, d[j],$$

where

$$a = \frac{(1-\alpha) + (p-j)\beta}{1+(p-r)\beta}, \qquad b = \frac{\alpha + (j-r)\beta}{1+(p-r)\beta}.$$

These coefficients satisfy

$$a + b = \frac{(1-\alpha) + (p-j)\beta + \alpha + (j-r)\beta}{1+(p-r)\beta} = 1.$$

Thus, every update is an affine combination of two points. For any affine transformation

$$T(x) = Ax + c,$$

we have

$$T(aP + bQ) = aT(P) + bT(Q),$$

whenever $a+b=1$. Therefore, applying an affine transformation to the control points and then evaluating the curve gives the same result as evaluating the curve first and then applying the affine transformation. Hence, the β-de Boor curve is affine invariant, provided that

$$1 + (p-r)\beta \neq 0.$$

Then, We prove the partition of unity property. Consider the update rule in the iterative β–de Boor algorithm:

$$d[j] \leftarrow \frac{(1-\alpha) + (p-j)\beta}{1+(p-r)\beta}\, d[j-1] + \frac{\alpha + (j-r)\beta}{1+(p-r)\beta}\, d[j].$$

Define the coefficients

$$A = \frac{(1-\alpha) + (p-j)\beta}{1+(p-r)\beta}, \qquad B = \frac{\alpha + (j-r)\beta}{1+(p-r)\beta}.$$

We compute their sum:

$$\begin{aligned} A + B &= \frac{(1-\alpha) + (p-j)\beta}{1+(p-r)\beta} + \frac{\alpha + (j-r)\beta}{1+(p-r)\beta} \\ &= \frac{(1-\alpha) + (p-j)\beta + \alpha + (j-r)\beta}{1+(p-r)\beta}. \end{aligned}$$

Simplifying the numerator,

$$(1-\alpha)+\alpha = 1, \qquad (p-j)\beta + (j-r)\beta = (p-r)\beta,$$

so that

$$A + B = \frac{1+(p-r)\beta}{1+(p-r)\beta} = 1.$$

Therefore, each update step is an affine combination of $d[j-1]$ and $d[j]$. By induction over the recursion, the final result $d[p]$ is an affine combination of the original control points. Consequently, the associated basis functions satisfy the partition of unity property,

$$\sum_i N_{i,p}^{(\beta)}(u) = 1.$$

Now, we can prove the convex hull property. The coefficients must satisfy

$$A \geq 0, \qquad B \geq 0, \qquad \text{and} \qquad 1+(p-r)\beta > 0.$$

From the definitions

$$A = \frac{(1-\alpha)+(p-j)\beta}{1+(p-r)\beta}, \qquad B = \frac{\alpha+(j-r)\beta}{1+(p-r)\beta},$$

these conditions are equivalent to

$$(1-\alpha)+(p-j)\beta \geq 0, \qquad \alpha+(j-r)\beta \geq 0, \qquad 1+(p-r)\beta > 0.$$

In the standard setting, the parameter α satisfies

$$0 \leq \alpha \leq 1.$$

Therefore, if $\beta \geq 0$, all three inequalities are satisfied, and hence

$$A \geq 0, \qquad B \geq 0.$$

Thus, a sufficient condition for non-negativity of the coefficients is

$$\boxed{\beta \geq 0.}$$

Then, we prove the endpoint interpolation property. Let a B-spline curve be evaluated using the iterative β–de Boor algorithm:

$$d[j] \leftarrow c[j+k-p], \quad j = 0, \ldots, p,$$

followed by the recursion

$$d[j] \leftarrow \frac{(1-\alpha)+(p-j)\beta}{1+(p-r)\beta} d[j-1] + \frac{\alpha+(j-r)\beta}{1+(p-r)\beta} d[j],$$

for $r = 1, \ldots, p$ and $j = p, \ldots, r$, where

$$\alpha = \frac{x - t[j+k-p]}{t[j+k-r+1] - t[j+k-p]}.$$

Assume a clamped knot vector:

$$t_0 = \cdots = t_p, \qquad t_{n+1} = \cdots = t_{n+p+1}.$$

We prove that the curve interpolates the endpoints:

$$C(t_p) = P_0, \qquad C(t_{n+1}) = P_n.$$

Left Endpoint: $C(t_p) = P_0$

Set $x = t_p$ and $k = p$. Then initialization gives

$$d[j] = c[j], \quad j = 0, \ldots, p.$$

For all $j \in \{r, \ldots, p\}$, we have $t[j] = t_p$, so

$$\alpha = \frac{t_p - t[j]}{t[j+p-r+1] - t[j]} = 0.$$

Thus the update becomes

$$d[j] \leftarrow \frac{1 + (p-j)\beta}{1 + (p-r)\beta} d[j-1] + \frac{(j-r)\beta}{1 + (p-r)\beta} d[j].$$

At $j = r$,

$$\frac{1 + (p-r)\beta}{1 + (p-r)\beta} = 1, \qquad \frac{0}{1 + (p-r)\beta} = 0,$$

so

$$d[r] \leftarrow d[r-1].$$

We prove by induction that $d[r] = c[0]$ for all r.
Base case ($r = 1$):

$$d[1] \leftarrow d[0] = c[0].$$

Inductive step: Assume $d[r-1] = c[0]$. Then

$$d[r] \leftarrow d[r-1] = c[0].$$

Thus $d[p] = c[0] = P_0$, and

$$\boxed{C(t_p) = P_0.}$$

Right Endpoint: $C(t_{n+1}) = P_n$

Set $x = t_{n+1}$ and $k = n$. Initialization gives

$$d[j] = c[j+n-p], \quad j = 0, \ldots, p,$$

so in particular $d[p] = c[n]$.
For all $j \in \{r, \ldots, p\}$, we have

$$t[j+n-r+1] = t_{n+1},$$

so

$$\alpha = \frac{t_{n+1} - t[j+n-p]}{t_{n+1} - t[j+n-p]} = 1.$$

Thus the update becomes

$$d[j] \leftarrow \frac{(p-j)\beta}{1 + (p-r)\beta} d[j-1] + \frac{1 + (j-r)\beta}{1 + (p-r)\beta} d[j].$$

At $j = p$,

$$\frac{0}{1+(p-r)\beta} = 0, \qquad \frac{1+(p-r)\beta}{1+(p-r)\beta} = 1,$$

so

$$d[p] \leftarrow d[p].$$

Thus $d[p]$ remains unchanged throughout the algorithm, and since initially

$$d[p] = c[n],$$

we obtain

$$\boxed{C(t_{n+1}) = P_n.}$$

For the β–de Boor algorithm with a clamped knot vector,

$$\boxed{C(t_p) = P_0, \qquad C(t_{n+1}) = P_n.}$$

In general, the modified curve does *not* satisfy the standard B-spline continuity property

$$C(u) \in C^{p-k}$$

at a knot of multiplicity k. This can be shown by a counterexample. Consider the quadratic case $p = 2$ with knot vector

$$U = \{0, 0, 0, 1, 2, 2, 2\}.$$

The interior knot $u = 1$ has multiplicity $k = 1$, so the standard B-spline result would predict $C^{p-k} = C^1$ continuity.

Using the modified de Boor recursion, the left and right limits at $u = 1$ are

$$C(1^-) = \frac{P_1 + (1+2\beta)P_2}{2(1+\beta)}$$

and

$$C(1^+) = \frac{(1+2\beta)P_1 + P_2}{2(1+\beta)}.$$

Therefore,

$$C(1^-) - C(1^+) = \frac{P_1 + (1+2\beta)P_2 - (1+2\beta)P_1 - P_2}{2(1+\beta)}.$$

Simplifying gives

$$C(1^-) - C(1^+) = \frac{2\beta(P_2 - P_1)}{2(1+\beta)} = \frac{\beta(P_2 - P_1)}{1+\beta}.$$

Thus, unless

$$\beta = 0 \qquad \text{or} \qquad P_1 = P_2,$$

we have

$$C(1^-) \neq C(1^+).$$

Hence the curve is not even C^0 continuous at the simple interior knot. Therefore, for $\beta \neq 0$, the usual result

$$C(u) \in C^{p-k}$$

does not hold in general. The classical continuity result is recovered only in the case $\beta = 0$, where the modified recursion reduces to the standard de Boor algorithm.

If adjacent curve segments share no common control points except at the joints, then the curve behaves like a piecewise Bézier curve. Suppose two neighboring segments are denoted by $C_i(u)$ and $C_{i+1}(u)$, and suppose they share the joint control point

$$P_i^{\text{end}} = P_{i+1}^{\text{start}}.$$

Then the endpoint values match, so

$$C_i(1) = C_{i+1}(0),$$

which gives C^0 continuity at the joint.

However, higher-order continuity is not automatic. For C^1 continuity, the tangent vectors must also agree:

$$C_i'(1) = C_{i+1}'(0).$$

This typically requires an additional constraint on the neighboring control points, such as

$$P_i^{\text{end}} - P_i^{\text{prev}} = P_{i+1}^{\text{next}} - P_{i+1}^{\text{start}},$$

possibly with a scaling factor depending on the parameterization. For C^2 continuity, the second derivatives must also match:

$$C_i''(1) = C_{i+1}''(0),$$

which imposes further constraints on the surrounding control points.

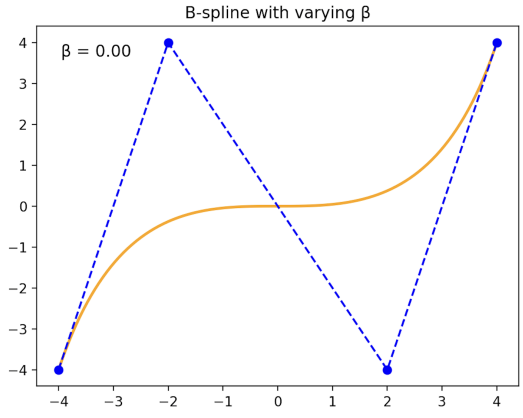
Therefore, if neighboring segments share only the endpoint control point, then the curve is generally C^0 continuous, but it is not necessarily C^1, C^2, or higher. Hence the standard B-spline result

$$C(u) \in C^{p-k}$$

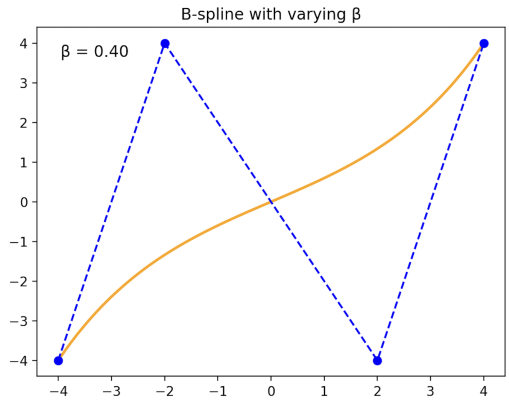
does not follow merely from sharing the joint point. Higher-order continuity must be enforced by additional geometric constraints between neighboring segments.

3.1 RESULTS

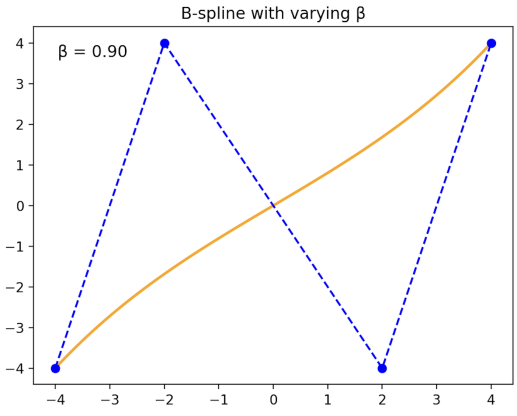
One way to adjust β is to use a slider. The following figures show a Bézier curve at different values of β.



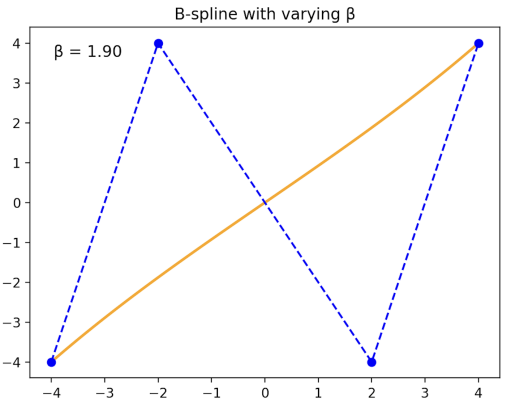
(a) β-B-Spline curve with knot vector (0 0 0 0 1 1 1 1) (AKA a β-Bezier Curve)



(b) β-B-Spline curve with knot vector (0 0 0 0 1 1 1 1) (AKA a β-Bezier Curve)



(c) β-B-Spline curve with knot vector (0 0 0 0 1 1 1 1) (AKA a β-Bezier Curve)



(d) β-B-Spline curve with knot vector (0 0 0 0 1 1 1 1) (AKA a β-Bezier Curve)

Figure 5: A Bézier curve at different values of β.

As can be seen, increasing β flattens the curve. The parameter β affects the curvature indirectly by modifying the blending functions of the curve. We may write the curve as

$$C_\beta(u) = \sum_i N_{i,p}^{(\beta)}(u) P_i,$$

where $N_{i,p}^{(\beta)}(u)$ denotes the modified basis function depending on β. The curvature of a spatial curve is

$$\kappa_\beta(u) = \frac{\left\| C'_\beta(u) \times C''_\beta(u) \right\|}{\left\| C'_\beta(u) \right\|^3},$$

and for a planar curve it is

$$\kappa_\beta(u) = \frac{\left| x'_\beta(u) y''_\beta(u) - y'_\beta(u) x''_\beta(u) \right|}{\left((x'_\beta(u))^2 + (y'_\beta(u))^2 \right)^{3/2}}.$$

Since β changes the basis functions $N_{i,p}^{(\beta)}(u)$, it also changes both the first and second derivatives,

$$C'_\beta(u) \qquad \text{and} \qquad C''_\beta(u).$$

Therefore, the curvature generally changes with β.

In the modified de Boor recursion, the blending coefficients have the form

$$a = \frac{(1-\alpha) + (p-j)\beta}{1 + (p-r)\beta}, \qquad b = \frac{\alpha + (j-r)\beta}{1 + (p-r)\beta}.$$

When $\beta > 0$, the denominator $1 + (p-r)\beta$ increases, so the dependence of the coefficients on the local parameter α is weakened. Geometrically, this often makes the curve more averaged or smoother-looking, which may reduce sharp curvature changes.

However, curvature is a nonlinear ratio involving both $C'_\beta(u)$ and $C''_\beta(u)$. Therefore, there is no universal guarantee that positive β always decreases curvature. Depending on the control points, increasing β may decrease curvature in some regions and increase it in others. Hence the safest statement is

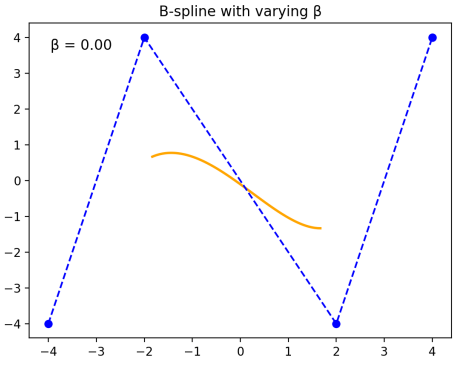
β changes the curvature by modifying the first and second derivatives of the curve.

For $\beta = 0$, the curve reduces to the standard B-spline and

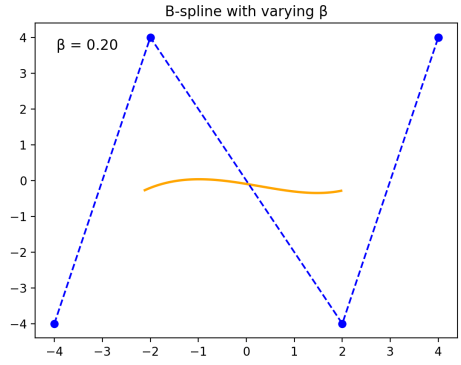
$$\kappa_\beta(u) = \kappa_0(u).$$

For $\beta \neq 0$, the curvature is generally different and depends on both β and the control polygon. The following figures show a B-Spline curve at different values of β.

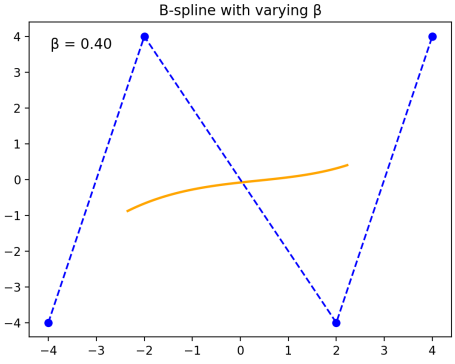
In addition to the curve flattening out, the endpoints of the curve approach the endpoints of the control polygons in the limit as β tends to infinity.



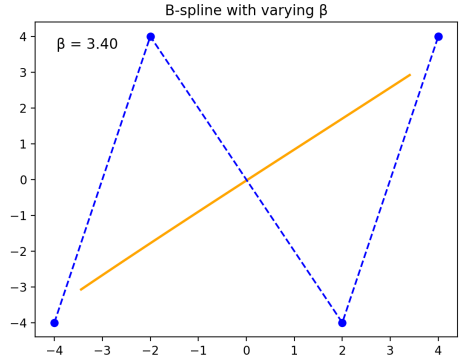
(a) β-B-Spline curve with knot vector (0 0 0 0.25 0.5 0.75 1 1 1)



(b) β-B-Spline curve with knot vector (0 0 0 0.25 0.5 0.75 1 1 1



(c) β-B-Spline curve with knot vector (0 0 0 0.25 0.5 0.75 1 1 1



(d) β-B-Spline curve with knot vector (0 0 0 0.25 0.5 0.75 1 1 1

Figure 6: A B-Spline curve at different values of β.

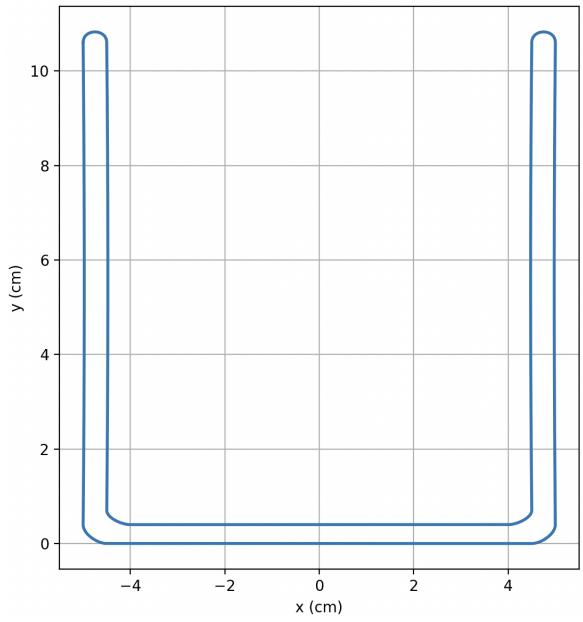


Figure 7

Figures 7-10 show a more involved shape with different values of β.

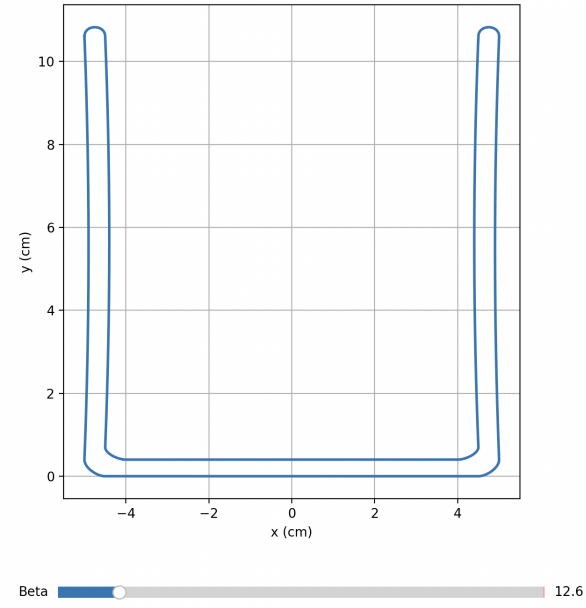


Figure 8

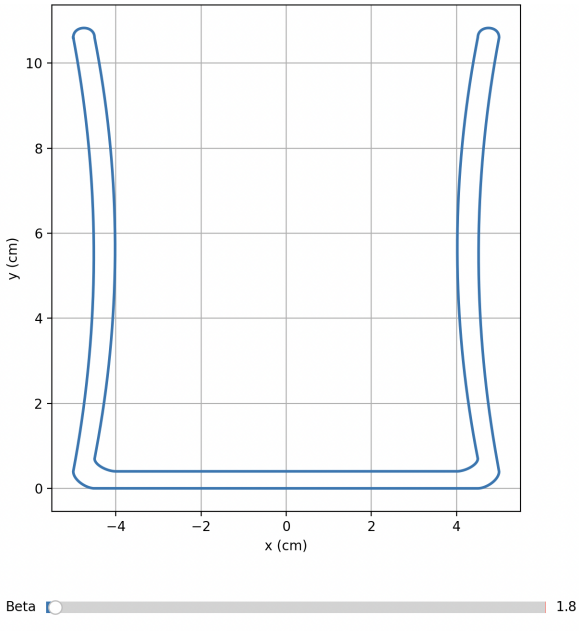


Figure 9

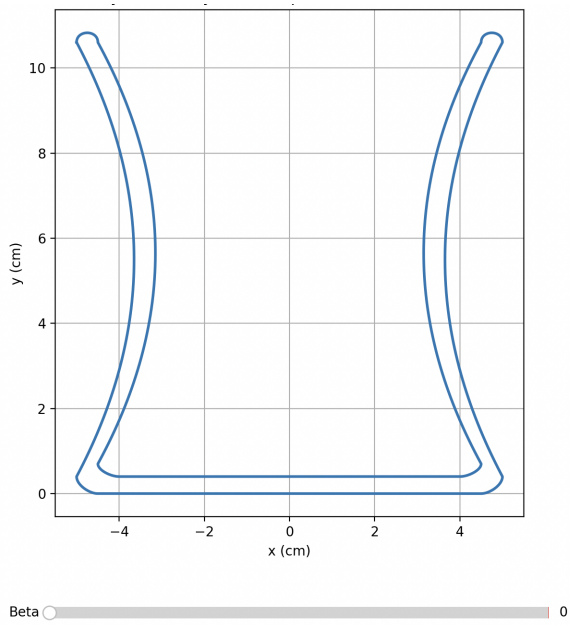


Figure 10

3.2 Side-Only Morphing by Decreasing β

A U-shaped object can be built first by constructing a composite cubic Bézier curve and then converting it to a cubic β-B-Spline curve. The shape (figure 11) has control points

"A": [(-4.5, 0.0), (-1.5, 0.0), (1.5, 0.0), (4.5, 0.0)],
"B": [(-4.0, 0.4), (-1.33, 0.4), (1.33, 0.4), (4.0, 0.4)],
"C": [(5.0, 0.4), (5.0, 3.8), (5.0, 7.2), (5.0, 10.6)],
"D": [(-5.0, 0.4), (-5.0, 3.8), (-5.0, 7.2), (-5.0, 10.6)],
"E": [(4.5, 0.0), (4.7, 0.0), (5.0, 0.2), (5.0, 0.4)],
"F": [(-4.5, 0.0), (-4.7, 0.0), (-5.0, 0.2), (-5.0, 0.4)],
"G": [(4.5, 0.7), (4.5, 4.0), (4.5, 7.3), (4.5, 10.6)],
"H": [(-4.5, 0.7), (-4.5, 4.0), (-4.5, 7.3), (-4.5, 10.6)],
"I": [(4.0, 0.4), (4.2, 0.4), (4.5, 0.55), (4.5, 0.7)],
"J": [(-4.0, 0.4), (-4.2, 0.4), (-4.5, 0.55), (-4.5, 0.7)],
"K": [(4.5, 10.6), (4.5, 10.9), (5.0, 10.9), (5.0, 10.6)],
"L": [(-5.0, 10.6), (-5.0, 10.9), (-4.5, 10.9), (-4.5, 10.6)].

Its pieces consist of two base curves A and B, four corner curves E, F, I and J, two top curves K and L, and four side curves C, D, G, and H. Let

$$\mathcal{S}_{\text{side}} = \{C, D, G, H\}$$

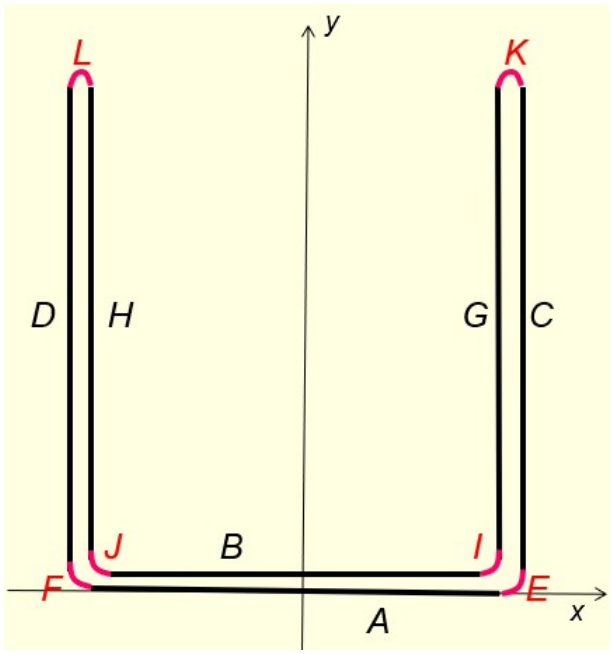


Figure 11

denote the set of side curves. The morphing method is applied only to these side curves. All other curve pieces remain fixed.

The morphing parameter is β itself. No additional animation parameter is introduced. The morph is obtained by evaluating the side curves at decreasing values of β. For each value of β, the side-curve control points are recomputed by the β-dependent morphing equations.

For a cubic β-Bézier segment, the basis functions are

$$B_k^3(t;\beta) = \binom{3}{k} \frac{\prod_{i=0}^{k-1}(t+i\beta)\prod_{j=0}^{2-k}((1-t)+j\beta)}{\prod_{m=0}^{2}(1+m\beta)}, \qquad k=0,1,2,3.$$

Thus a cubic segment can be written as

$$C(t;\beta) = \sum_{k=0}^{3} P_k B_k^3(t;\beta), \qquad 0 \le t \le 1.$$

For a local clamped knot vector

$$\Xi = \{0,0,0,0,1,1,1,1\},$$

this segment can also be interpreted as a cubic β-B-spline segment.

Let

$$U(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}.$$

By expanding the basis functions $B_k^3(t;\beta)$, the same segment can be written in matrix form as

$$C(t;\beta) = U(t)M_\beta \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}.$$

Here M_β is the cubic coefficient matrix obtained by expanding the β-basis functions in powers of t. When $\beta = 0$, the β-basis reduces to the ordinary cubic Bernstein basis, and hence M_β reduces to the ordinary cubic matrix M_0.

For each side curve $s \in \mathcal{S}_{\text{side}}$, the data points remain fixed, but the internal control points are recalculated for each value of β. Suppose a side curve is interpolated through data points

$$X_0^{(s)}, X_1^{(s)}, \ldots, X_n^{(s)}.$$

The i-th cubic piece of side curve s has the form

$$C_i^{(s)}(t;\beta) = \sum_{k=0}^{3} P_{i,k}^{(s)} B_k^3(t;\beta), \qquad 0 \le t \le 1.$$

The endpoint interpolation conditions are

$$P_{i,0}^{(s)}(\beta) = X_i^{(s)}, \qquad P_{i,3}^{(s)}(\beta) = X_{i+1}^{(s)}.$$

The two interior control points are denoted by

$$P_{i,1}^{(s)}(\beta) = A_i^{(s)}(\beta), \qquad P_{i,2}^{(s)}(\beta) = B_i^{(s)}(\beta).$$

Therefore,

$$C_i^{(s)}(t;\beta) = X_i^{(s)} B_0^3(t;\beta) + A_i^{(s)}(\beta) B_1^3(t;\beta) + B_i^{(s)}(\beta) B_2^3(t;\beta) + X_{i+1}^{(s)} B_3^3(t;\beta).$$

The morphing method recomputes the unknown interior control points $A_i^{(s)}(\beta)$ and $B_i^{(s)}(\beta)$ by solving a β-dependent linear system. In matrix form, this system is written as

$$\mathcal{M}^{(s)}(\beta)\mathbf{z}^{(s)}(\beta) = \mathbf{r}^{(s)}(\beta),$$

where

$$\mathbf{z}^{(s)}(\beta) = \begin{bmatrix} A_0^{(s)}(\beta) \\ B_0^{(s)}(\beta) \\ A_1^{(s)}(\beta) \\ B_1^{(s)}(\beta) \\ \vdots \\ A_{n-1}^{(s)}(\beta) \\ B_{n-1}^{(s)}(\beta) \end{bmatrix}.$$

Hence

$$\mathbf{z}^{(s)}(\beta) = \left(\mathcal{M}^{(s)}(\beta)\right)^{-1} \mathbf{r}^{(s)}(\beta).$$

This is the essential step of the morphing process: the side control points are recalculated as functions of β.

For non-side pieces, no morphing is applied. Therefore their control points remain fixed and they are evaluated with $\beta = 0$. Equivalently, for each curve piece s, define

$$\beta_s = \begin{cases} \beta, & s \in \mathcal{S}_{\text{side}}, \\ 0, & s \notin \mathcal{S}_{\text{side}}. \end{cases}$$

Thus only the side curves depend on the current value of β.

After the side control points have been recomputed, each local β-curve segment is converted into an equivalent ordinary cubic B-spline segment. For a given local segment, define

$$\mathbf{P}_i^{(s)}(\beta) = \begin{bmatrix} P_{i,0}^{(s)}(\beta) \\ P_{i,1}^{(s)}(\beta) \\ P_{i,2}^{(s)}(\beta) \\ P_{i,3}^{(s)}(\beta) \end{bmatrix}.$$

The β-segment is

$$C_i^{(s)}(t;\beta) = U(t) M_{\beta_s} \mathbf{P}_i^{(s)}(\beta).$$

We seek an ordinary cubic B-spline segment with control vector

$$\mathbf{Q}_i^{(s)}(\beta) = \begin{bmatrix} Q_{i,0}^{(s)}(\beta) \\ Q_{i,1}^{(s)}(\beta) \\ Q_{i,2}^{(s)}(\beta) \\ Q_{i,3}^{(s)}(\beta) \end{bmatrix}$$

such that

$$C_i^{(s)}(t;\beta) = U(t) M_0 \mathbf{Q}_i^{(s)}(\beta).$$

Setting the two polynomial forms equal gives

$$U(t) M_{\beta_s} \mathbf{P}_i^{(s)}(\beta) = U(t) M_0 \mathbf{Q}_i^{(s)}(\beta).$$

Since this equality must hold for all t, the polynomial coefficients must agree:

$$M_{\beta_s} \mathbf{P}_i^{(s)}(\beta) = M_0 \mathbf{Q}_i^{(s)}(\beta).$$

Solving for the equivalent ordinary control vector gives

$$\mathbf{Q}_i^{(s)}(\beta) = M_0^{-1} M_{\beta_s} \mathbf{P}_i^{(s)}(\beta).$$

For non-side pieces, $\beta_s = 0$, so

$$\mathbf{Q}_i^{(s)}(\beta) = M_0^{-1} M_0 \mathbf{P}_i^{(s)} = \mathbf{P}_i^{(s)}.$$

Hence the base, corner, and top pieces remain unchanged.

Finally, the converted curve is evaluated using the ordinary cubic B-spline equation

$$C_i^{(s)}(t;\beta) = \sum_{k=0}^{3} N_{k,3}(t) Q_{i,k}^{(s)}(\beta),$$

where $N_{k,3}(t)$ are the ordinary cubic B-spline basis functions over the local clamped knot vector

$$\Xi = \{0, 0, 0, 0, 1, 1, 1, 1\}.$$

The ordinary Cox–de Boor basis functions are defined recursively by

$$N_{i,0}(t) = \begin{cases} 1, & \xi_i \le t < \xi_{i+1}, \\ 0, & \text{otherwise}, \end{cases}$$

and

$$N_{i,p}(t) = \frac{t - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(t) + \frac{\xi_{i+p+1} - t}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(t),$$

where terms with zero denominators are omitted.

The full composite curve is therefore

$$C(t;\beta) = \bigcup_s \bigcup_i \sum_{k=0}^{3} N_{k,3}(t) Q_{i,k}^{(s)}(\beta).$$

Only the side curves C, D, G, H depend on β, because only their control points are recalculated by the morphing method. The base curves, corner curves, and top curves remain unchanged. Therefore the morphing is side-only and is controlled entirely by decreasing β.

4 CONCLUSIONS

This paper gives new definitions of B-Spline curves. It shows that the new type of curves not only has all the basic properties of its traditional predecessor but also can deform without its control points being manipulated. Therefore, we have a curve design technique more general than traditional B-Spline curves.

Future work in this direction includes studying β–B-Spline surfaces and extending the Beta shape parameter concept into subdivision surfaces.

REFERENCES

[1] Cheng, F.; Kazadi, A.; Lin, A.: Beta-bezier curves. CAD'20, 343–347, 2020. http://doi.org/10.14733/cadconfP.2020.343-347.

[2] Chu, L.; Zeng, X.: Constructing curves and triangular patches by beta functions. Journal of Computational and Applied Mathematics, 260, 191–200, 2014. http://doi.org/10.1016/j.cam.2013.09.025.

[3] Gallier, J.: Curves and surfaces in geometric modeling: Theory and algorithms. Morgan Kaufmann, 1999. http://doi.org/10.5555/320367.

[4] Han, X.A.; Ma, Y.; Huang, X.: A novel generalization of bézier curve and surface. J. Comput. Appl. Math., 217(1), 180–193, 2008. http://doi.org/10.1016/j.cam.2007.06.027.

[5] Kovács, I.; Várady, T.: P-bézier and p-bspline curves – new representations with proximity control. Computer Aided Geometric Design, 62, 117–132, 2018. http://doi.org/10.1016/j.cagd.2018.03.020.

[6] Levent, A.; Sahin, B.: Beta-bezier curves. Applied and Computational Mathematics, 18, 79–94, 2019.

[7] Prautzsch, H.; Boehm, W.: In Geometric Concepts for Geometric Design.

[8] Yan, L.; Liang, J.: An extension of the bezier model. Applied Mathematics and Computation, 218, 2863–2879, 2011. http://doi.org/10.1016/j.amc.2011.08.030.

[9] Yang, L.; Zeng, X.M.: Bézier curves and surfaces with shape parameters. International Journal of Computer Mathematics, 86(7), 12531263, 2009. http://doi.org/10.1080/00207160701821715.