Title:
Computer Aided Design by Adjusting Tension Parameters

Authors:
Seifalla Moustafa, seifalla.moustafa@uky.edu, University of Kentucky
Fuhua (Frank) Cheng, cheng@cs.uky.edu, University of Kentucky
Shuhua Lai, slai@ggc.edu, Georgia Gwinnett College
Alice J. Lin, lina@apsu.edu, Austin Peay State University
Anastasia Kazadi, anastasia.kazadi@eku.edu, Eastern Kentucky University

Keywords:
Bezier curves, B-Spline Curves, Beta-Bezier Curves, Beta-B-Spline, Tension Control, Interpolation

DOI: 10.14733/cadconfP.2026.146-153

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, 5, 6, 7]. 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.

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 carton/game figure design, surgery simulation, and TV special effects.

This paper presents β-Bézier and β-B-spline curves, which extend the classical Bézier and B-spline formulations by introducing a tension parameter β derived from the properties of the beta function. By adapting the De Casteljau and de Boor algorithms, 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.

Main Idea:

A famous function [1], in mathematics is the beta-function:

$$b(\alpha,\gamma)=\int_0^1 x^{\alpha-1}(1-x)^{\gamma-1}\,dx. \tag{2.1}$$

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:

$$b(\alpha+1,\gamma)=\frac{\alpha}{\alpha+\gamma}\,b(\alpha,\gamma), \tag{2.2}$$

$$b(\alpha,\gamma+1)=\frac{\gamma}{\alpha+\gamma}\,b(\alpha,\gamma). \tag{2.3}$$

Let $\alpha=\lambda t$ and $\gamma=\lambda(1-t)$. Applying (4) k times and then (5) $n-k$ times, we get

$$b(\lambda t+k,\lambda(1-t)+n-k)=\frac{\prod_{i=0}^{k-1}(\lambda t+i)\ \prod_{j=0}^{n-k-1}(\lambda(1-t)+j)}{\prod_{m=0}^{n-1}(\lambda+m)}\,b(\lambda t,\lambda(1-t)). \tag{2.4}$$

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:

$$B_k^n(t;\lambda)=\binom{n}{k}\frac{\prod_{i=0}^{k-1}(\lambda t+i)\ \prod_{j=0}^{n-k-1}(\lambda(1-t)+j)}{\prod_{m=0}^{n-1}(\lambda+m)}. \tag{2.5}$$

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

$$\lim_{\lambda\to\infty}\binom{n}{k}\frac{\prod_{i=0}^{k-1}(\lambda t+i)\ \prod_{j=0}^{n-k-1}(\lambda(1-t)+j)}{\prod_{m=0}^{n-1}(\lambda+m)}=\binom{n}{k}t^k(1-t)^{n-k} \tag{2.6}$$

which is the Bernstein polynomials used in the formula for a Bezier curve. This was developed by Zeng et al. [1].

It has been shown [2] that the properties of β–Bézier curves are easier to study when $\lambda=\frac{1}{\beta}$. In this case, we have

$$B_k^n(t;\beta)=\binom{n}{k}\frac{\prod_{i=0}^{k-1}(t+i\beta)\ \prod_{j=0}^{n-k-1}((1-t)+j\beta)}{\prod_{m=0}^{n-1}(1+m\beta)}. \tag{2.7}$$

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

$$C(t;\beta)=\sum_{k=0}^{n}P_k\,B_k^n(t;\beta) \tag{2.8}$$

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

1: **for** $i = 0$ to n **do**
2: $\quad P_i^0 \leftarrow P_i$
3: **end for**
4: **for** $i = 1$ to n **do**
5: $\quad$ **for** $j = i$ to n **do**
6: $\quad\quad P_j^i \leftarrow \frac{1-t+(n-j)\beta}{1+(n-i)\beta} P_{j-1}^{i-1} + \frac{t+(j-i)\beta}{1+(n-i)\beta} P_j^{i-1}$
7: $\quad$ **end for**
8: **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** $d[p]$

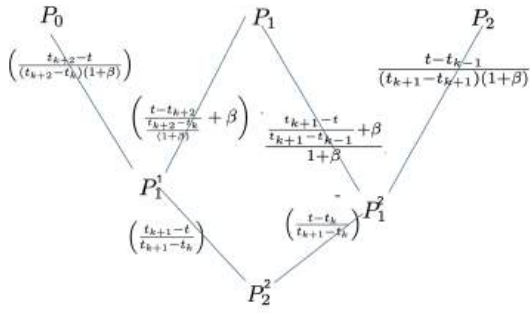


Fig. 1: Pascal Triangle for the Quadratic Case

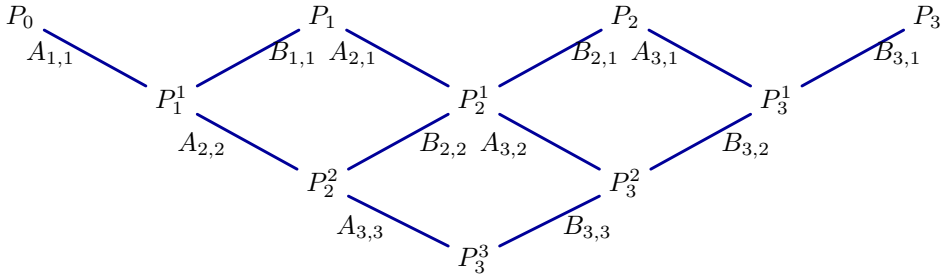
The iterative β-de Boor algorithm has a time complexity of $O(p^2)$, where p is the spline degree. The initialization loop executes $p+1$ times, while the nested loops perform

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

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.

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 1 and the below figures show derivations of the formulae for a cubic and a quadratic B-Spline curves:

$$\begin{aligned} P_2^2 = \frac{t_{k+1}-t}{t_{k+1}-t_k}\left(\frac{t_{k+2}-t}{(t_{k+2}-t_k)(1+\beta)}P_0 + \frac{\frac{t-t_k}{t_{k+2}-t_k}+\beta}{1+\beta}P_1\right) \\ + \frac{t-t_k}{t_{k+1}-t_k}\left(\frac{\frac{t_{k+1}-t}{t_{k+1}-t_{k-1}}+\beta}{1+\beta}P_1 + \frac{t-t_{k-1}}{(t_{k+1}-t_{k-1})(1+\beta)}P_2\right). \end{aligned} \tag{2.9}$$



$$P_3^3 = w_0\,P_0 + w_1\,P_1 + w_2\,P_2 + w_3\,P_3, \tag{2.10}$$

$$w_0 = A_{3,3}\, A_{2,2}\, A_{1,1}, \tag{2.11}$$

$$w_1 = A_{3,3}(A_{2,2}B_{1,1} + B_{2,2}A_{2,1}) + B_{3,3}\, A_{3,2}A_{2,1}, \tag{2.12}$$

$$w_2 = A_{3,3}\, B_{2,2}B_{2,1} + B_{3,3}(A_{3,2}B_{2,1} + B_{3,2}A_{3,1})\,, \tag{2.13}$$

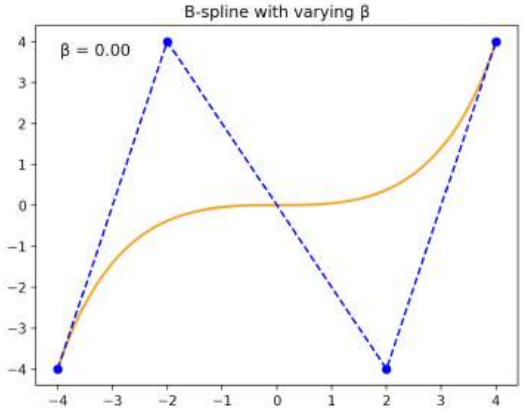
$$w_3 = B_{3,3}\, B_{3,2}\, B_{3,1}. \tag{2.14}$$

$$A_{j,r} = \frac{(1-\alpha_{j,r}) + (3-j)\beta}{1 + (3-r)\beta}, \tag{2.15}$$

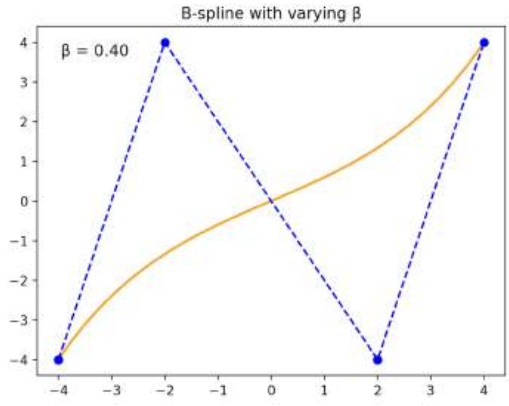
$$B_{j,r} = \frac{\alpha_{j,r} + (j-r)\beta}{1 + (3-r)\beta}, \tag{2.16}$$

$$\alpha_{j,r} = \frac{t - t_{j+k-3}}{t_{j+k-r+1} - t_{j+k-3}} \tag{2.17}$$

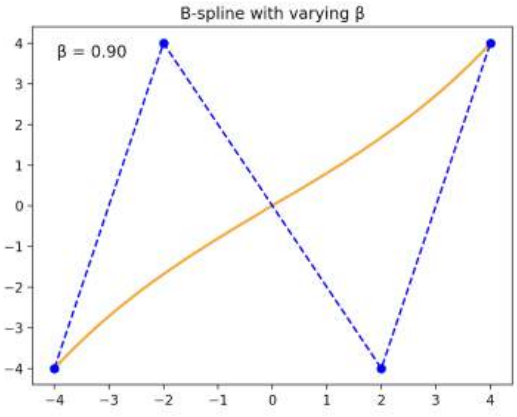
One way to adjust β is to use a slider. Below, we show examples of β-Bézier and β-B-Spline curves. The figures show how the shape of the curve changes with β.



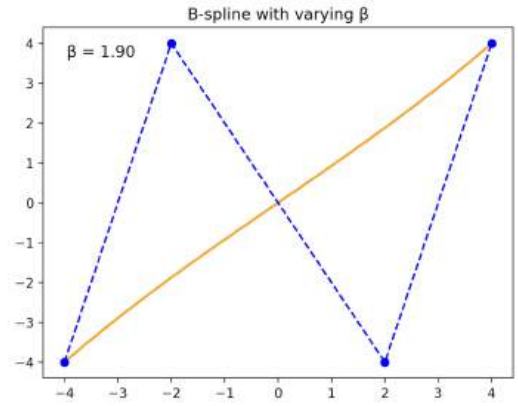
(a) β-B-Spline curve with knot vector (0 0 0 0 1 1 1 1) (AKA a β-Bezier Curve)



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

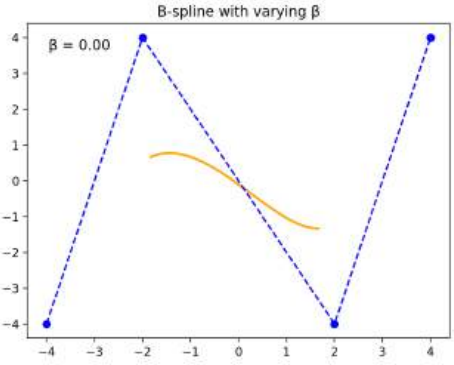


(c) β-B-Spline curve with knot vector (0 0 0 0 1 1 1 1) (AKA a β-Bezier Curve)

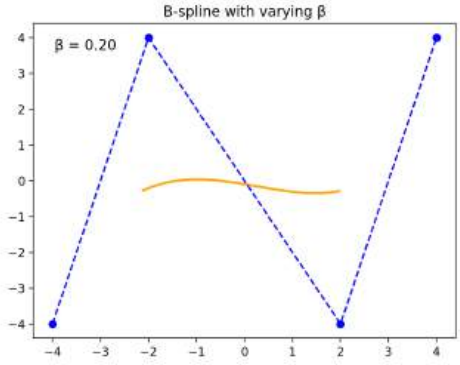


(d) β-B-Spline curve with knot vector (0 0 0 0 1 1 1 1) (AKA a β-Bezier Curve)

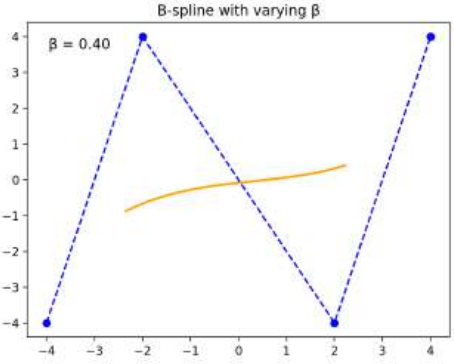
Fig. 2: Comparison of the four configurations.



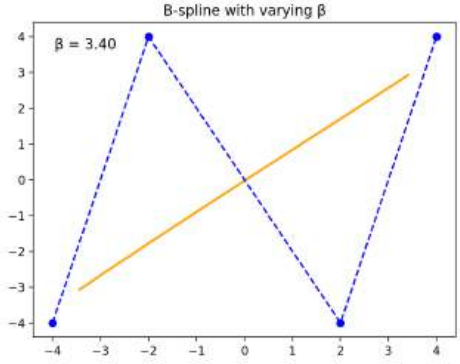
(a) β-B-Spline curve with knot vector (0 0 0 0.25 0.5 0.75 1 1 1)



(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

Fig. 3: Comparison of the four configurations.

Conclusions:

This paper gives new definitions of Bézier and B-Spline curves. It shows that the new types of curve not only have all the basic properties of their traditional predecessors but also can deform without their control points being manipulated. Therefore, we have a curve design technique more general than traditional Bézier and 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] L. Chu and XM. Zeng, *Constructing curves and triangular patches by beta functions*, Journal of Computational and Applied Mathematics, vol. 260, pp. 191–200, 2014.

[2] F. Cheng, A. Kazadi, and A. Lin, *Beta-Bezier curves*, CAD'20, pp. 343–347, 2020.

[3] J. Gallier, *Curves and Surfaces in Geometric Modeling: Theory and Algorithms*, Morgan Kaufmann 1999.

[4] XA. Han, Y. Ma, and X. Huang, *A novel generalization of Bezier curve and surface*, Journal of Computational and Applied Mathematics, pp. 180–193, 2008.

[5] A. Levent, B. Sahin, *Beta-Bezier curves*, Applied and Computational Mathematics, vol. 18, pp. 79–94, 2019.

[6] H. Prautzsch, and W. Boehm, *Geometric Concepts for Geometric Design*, Peters/CRC Press,

[7] L.L. Yan, and J.F. Liang, *An extension of the Bezier model*, Applied Mathematics and Computation, vol. 218, pp. 2863–2879.

[8] Y. Zhu, and X. Han, *Quasi-Bernstein-Bezier polynomials over triangular domain with multiple shape parameters*, Applied Mathematics and Computation, vol. 250, pp. 181–192, 2015.