Investigation on mathematical method used to construct curves in computer systems

Daekun Kim

April 28, 2018

Word count: 1447

Contents

1 Introduction

In the modern era, computer is one of the tools that are having the most influence in our lives. It is commonly used to design different products, such as cars and airplanes, or in the movie industry, such as producing scenes with computer graphics and animations. In these fields, ability to form and display smooth, realistic curvatures is essential. However, computers are digital, and they can only process discrete information. This nature of computers makes it seem to be only able to render discrete and pixelated shapes that cannot imitate natural curves. Yet, they seem to be able to easily generate many kinds of curved lines and surfaces, as shown in many computer-generated animations and products produced by computer-controlled machineries. This provoked my interest in how mathematics may play a role in achieving such seemingly impossible task.

This paper aims to investigate the mathematics behind how a computer system generates smooth curves. One of the most prominent methods used to achieve this is the *Bézier curve.* It is used in many popular photo-editing and animating softwares, such as the Adobe Flash®, Blender^{TM}, Autodesk Maya®, and more (*Bezier curves*, 2017; *Bezier Pen Tool - Flash Professional CS5*, 2017). This paper aims to clarify the journey that it took to come into being and be widespread to be utilized in numerous fields. Specifically, this paper describes the mathematical development of the Bézier curve, which starts from a rudimentary linear interpolation, repeated linear interpolation, the De Castaljau Algorithm, and finally the Bézier curve itself. Then, the properties of the Bézier curve that made it to be one of the most popular tool in curvature rendering in computer systems will be investigated.

2 Linear Interpolation

Given two points, (x_0, y_0) and (x_1, y_1) with $x_0 < x_1$, a *linear interpolation* is construction of a new set of y from x values within $[x_0, x_1]$. The formula of linear interpolation can be given as the relationship of the slope of the line connecting the starting and end point with that of line connecting the interpolated point with the starting point, or

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \tag{1}$$

Fig. 2.1 shows an example of linear interpolation between $(1, 1)$ and $(3, 2)$.

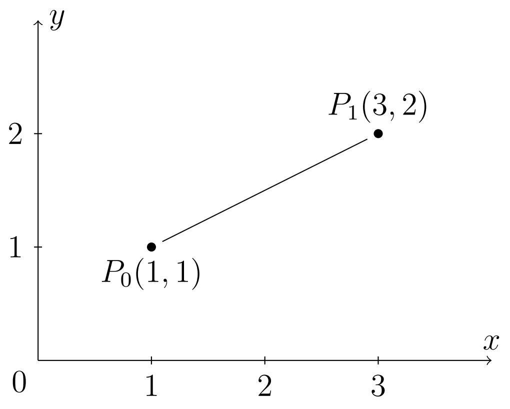

Figure 2.1: Linear interpolation between two points

The equation of linear interpolation in Fig. 2.1 can be defined as following:

$$\frac{y-1}{x-1} = \frac{2-1}{3-1}, 1 \leq x \leq 2 \tag{2}$$

Eq. (2) is stating that the slope of the line between (1, 1) and (3, 2), and any point along the interpolated line (called the *interpolant*) is the same. Solving this for x gives us the following:

$$\begin{aligned} y &= \frac{1}{2}(x-1) + 1, \\ y &= \frac{1}{2}x + \frac{1}{2} \end{aligned} \tag{3}$$

In fact, this is now in slope-intercept form of an equation of a linear function, $y = mx + b$. This signifies that linear interpolation is a linear function with a restricted domain between two points.

Instead of defining an interpolation in respect to x, we can do so with another variable t within $[0, 1]$ and form a parametric equation of this linear interpolation. The example above can be used to demonstrate this, as shown in Fig. 2.2.

Fig. 2.2 shows a linear interpolation at $t = \frac{1}{2}$. At $t = \frac{1}{2}$, the resulting point at halfway between P_0 and P_1 in both directions of the x- and y-axes. In the same way, at $t = \frac{1}{4}$ and $t = \frac{3}{4}$, the resulting points are one-quarter and three quarters of the way, respectively, between the two end points in both axes. This t value is essentially

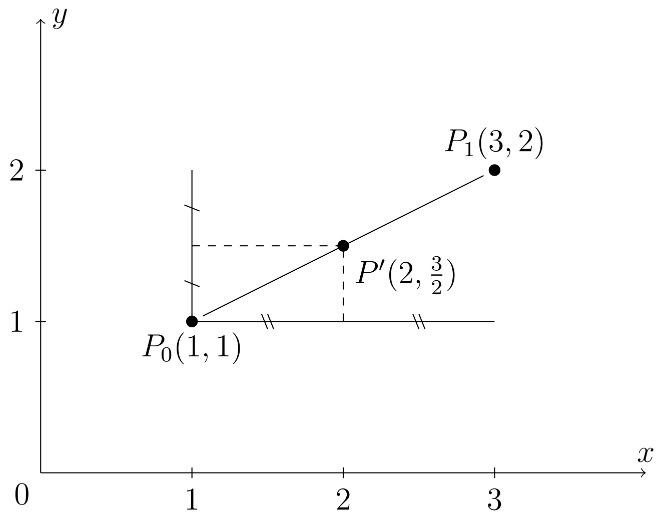

Figure 2.2: Linear interpolation between two points

the ratio that describes how far the interpolated point is from the end, just as a GPS device would let one know the percentage in which the person has driven from the starting point. As such, a linear interpolation between (x_0, y_0) and (x_1, y_1) can be parametrically defined as following:

$$x(t) = x_0 + (x_1 - x_0)t \tag{4}$$

$$y(t) = y_0 + (y_1 - y_0)t \tag{5}$$

Note that at $t = 0$, the interpolated point is at (x_0, y_0), the starting point, and at $t = 0$, it is at (x_1, y_1), the end point.

Let P_0 and P_1 be position vectors, where $P_0 = (x_0, y_0)$ and $P_1 = (x_1, y_1)$. We can now combine eq. (4) and (5) into one vector equation:

$$P(t) = P_0 + (P_1 - P_0)t, \tag{6}$$

where $P(t)$ is a function that linearly interpolates a point between P_0 and P_1 for $t \in [0, 1]$. Eq. (6) can be rearranged as following:

$$P(t) = P_0(1 - t) + P_1 t, \tag{7}$$

Eq. (7) further emphasizes the analogous nature of t being the “ratio” of the resulting point P with respect to P_0 and P_1 (Prautzsch, Boehm, & Paluszny, 2013). More specifically, the ratio $(1 - t) : t$ equals to the ratio of the distance of P from P_0 and P_1, respectively.

3 Recursive Linear Interpolation and De Casteljau Algorithm

Thus far, we have investigated on the method of producing a straight line with linear interpolation. This section will study how linear interpolation can be recursively repeated to produce a smooth curve.

3.1 EXAMPLE 1: Recursive Linear Interpolation with 3 Points

Let $P_0 = (1, 1)$, $P_1 = (2, 2)$ and $P_2 = (3, 1)$. These points are commonly referred to as the *control points* of the interpolated curve (Prautzsch et al., 2013). We can easily perform linear interpolations between P_0 and P_1, and between P_1 and P_2, as shown in Fig 3.1.

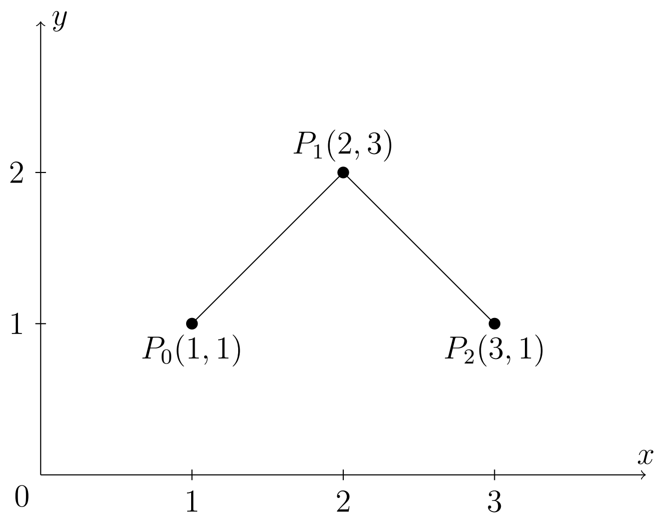

Figure 3.1: Linear interpolation between three points

Let $P_{0,1}$ be the linearly interpolated point between P_0 and P_1 at $t = 0.5$. Similarly, let $P_{1,2}$ be the linearly interpolated point between P_1 and P_2 at $t = 0.5$. The two newly interpolated points will be at the half point of their interpolants. Between these new points, another linear interpolation can be conducted, as shown in Fig. 3.2 with a dashed line.

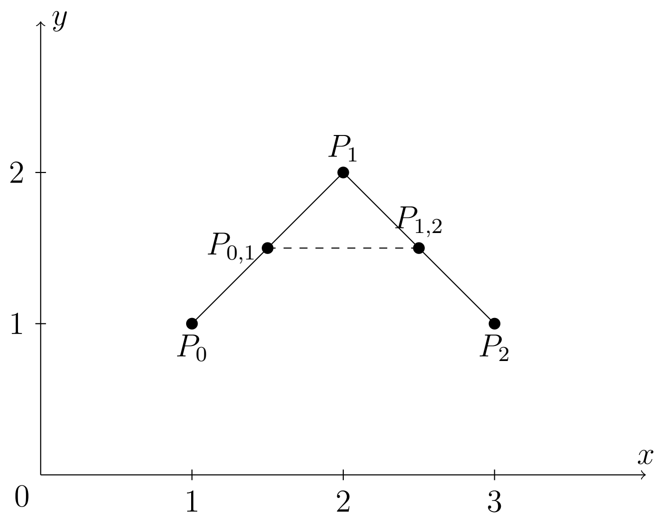

Figure 3.2: Visual representation of recursive linear approximation

Finally, let P' be the linearly interpolated point between $P_{0,1}$ and $P_{2,3}$ at $t = 0.5$. P', then, is the point on the interpolated curve of the control points P_0, P_1, and P_2 at $t = 0.5$, as shown in Fig. 3.3. Repeating this technique for $t \in [0, 1]$ allows one to plot the interpolated curve, as shown in Fig. 3.4.

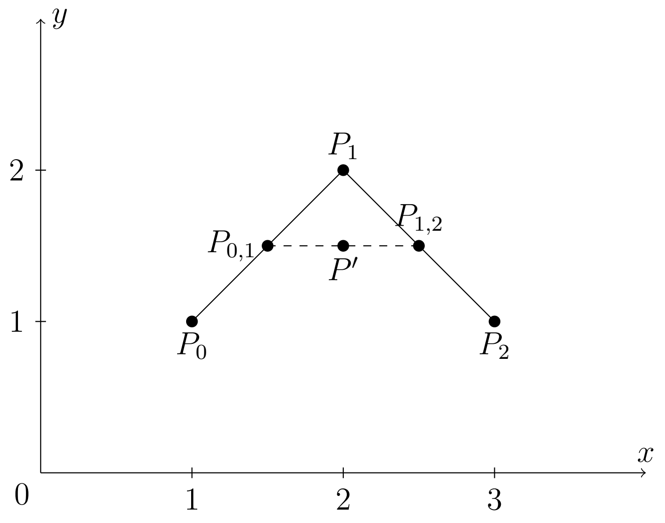

Figure 3.3: Interpolated point at $t = 0.5$

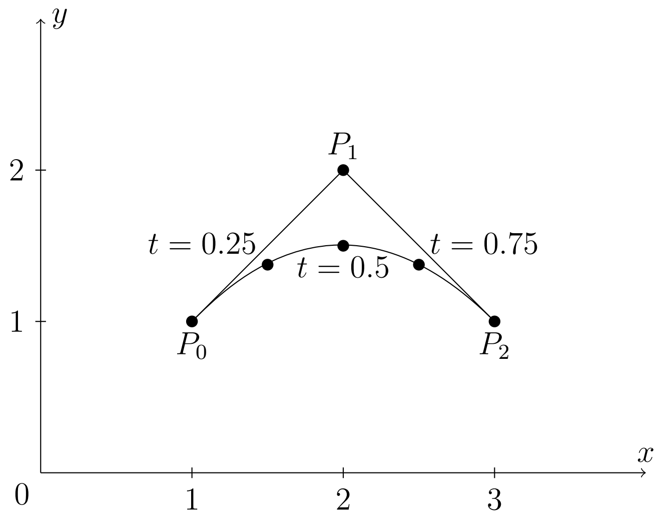

Figure 3.4: Illustration of the interpolated curve

This recursively repeated linear interpolation is called the De Casteljau construction (de Casteljau, 1959). This is commonly used to easily evaluate and illustrate a Bézier curve. The details of the Bézier curve itself will be discussed in the latter part of this essay, and instead, this section will be investigating how a number of rudimentary mathematical ideas led to the establishment of the concept of Bézier curve and deduce its properties.

The De Casteljau construction can be performed with higher number of control points.

3.2 EXAMPLE 2: Recursive Linear Interpolation with 4 Points

In this example, we will find the interpolated point at $t = 0.4$. Let $P_0 = (1, 1)$, $P_1 = (2, 3)$, $P_2 = (3, 3)$ and $P_3 = (4, 1)$.

First, we perform linear interpolation between each of the consecutive points. Then, the new interpolated point at $t = 0.4$ can be produced for each of the linear interpolations. We now repeat the same technique for the newly obtained points, until we are left with one single point, P'. Fig. 3.5 gives a visual illustration of this recursive construction.

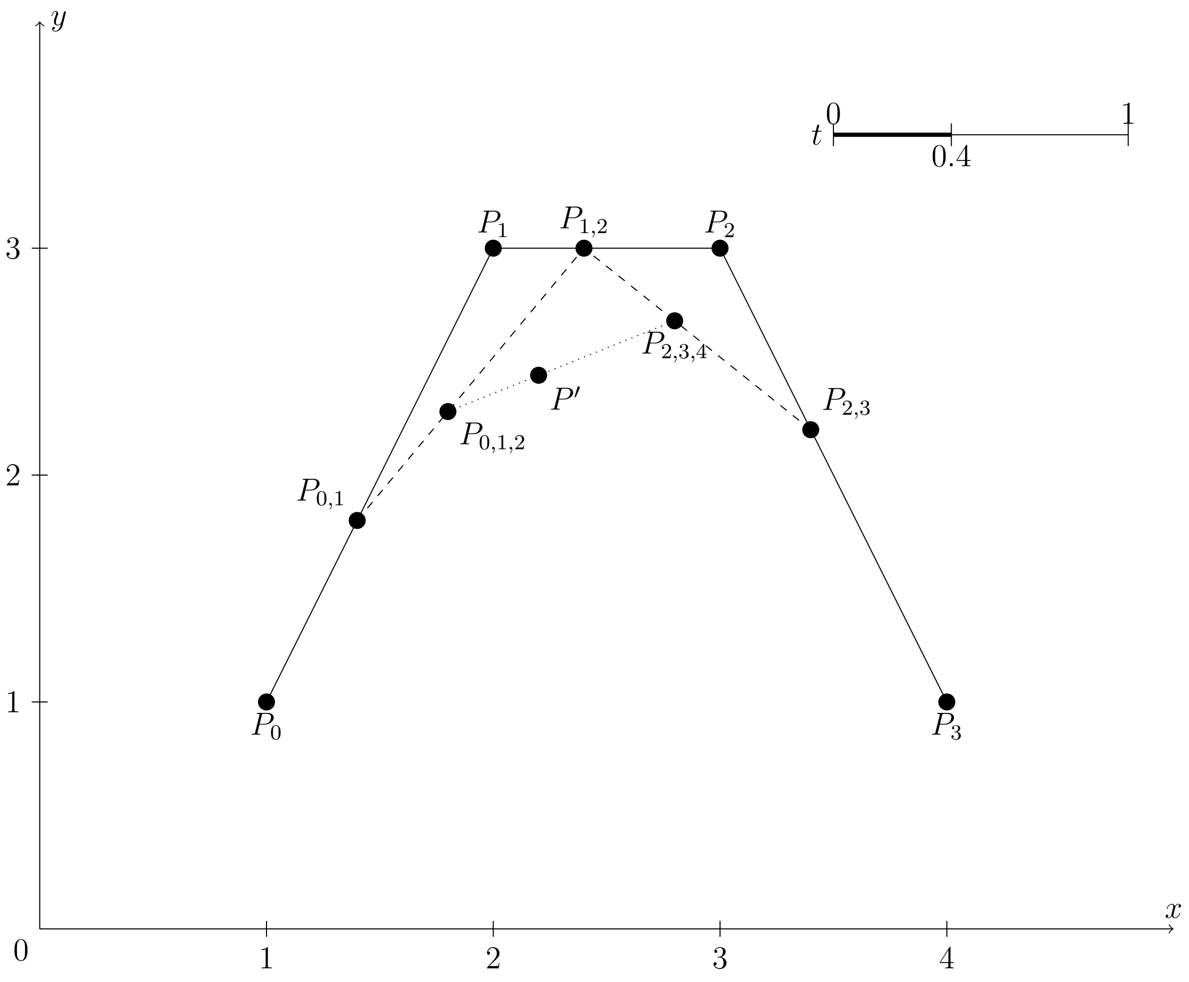

Figure 3.5: Illustration of the interpolated curve with 4 control points

3.3 The De Casteljau Algorithm

The two aforementioned examples allows a generalized set of procedures for the De Casteljau construction to be deduced. Given $n+1$ control points $P_0, P_1, P_2, \dots P_{n-1}, Pn$, an interpolation at $t \in [0, 1]$ can be obtained as following:

1. Create a new set of linearly interpolated points at t for each of the consecutive pair of control points.
2. Repeat #1 for newly interpolated points, until only a single point is left.

Note how this algorithm repeats itself until the objective it met. In other words, this algorithm is defined in terms of itself, meaning that this is a recursive procedure. The De Casteljau algorithm can now be expressed in a form of a mathematical equation.

Let $P_{a\ldots b}(t)$ be a function that gives an interpolated point from a set of control points $P_a, P_{a+1}, P_{a+2}, \ldots P_{b-1}, P_b$, where $a, b \in \mathbb{Z}_{\geq 0}$ and $a < b$. Also, let the given control points be $P_0, P_1, P_2, \ldots P_{n-1}, Pn$. By using the De Casteljau algorithm and the formula of linear interpolation from Eq. (7) we can derive the generalized formula to express the interpolation of a curve.

$$P_{0\ldots1}(t) = P_0(1-t) + P_1 t$$

$$P_{1\ldots2}(t) = P_1(1-t) + P_2 t$$

$$\vdots$$

$$P_{n-1\ldots n}(t) = P_{n-1}(1-t) + P_n t$$

We can now use the newly created linear interpolation functions to perform another set of linear interpolation.

$$P_{0\ldots2}(t) = [P_{0\ldots1}(t)](1-t) + [P_{1\ldots2}(t)]t$$

$$P_{1\ldots3}(t) = [P_{1\ldots2}(t)](1-t) + [P_{2\ldots3}(t)]t$$

$$\vdots$$

$$P_{n-2\ldots n}(t) = [P_{n-2\ldots n-1}(t)](1-t) + [P_{n-1\ldots n}(t)](t)t$$

The linear interpolation functions in the right-hand side of the equations can be substituted with their original function definitions.

$$P_{0\ldots2}(t) = [P_0(1-t) + P_1t](1-t) + [P_1(1-t) + P_2t]t$$

$$P_{1\ldots3}(t) = [P_1(1-t) + P_2t](1-t) + [P_2(1-t) + P_3t]t$$

$$\vdots$$

$$P_{n-2\ldots n}(t) = [P_{n-2}(1-t) + P_{n-1}t](1-t) + [P_{n-1}(1-t) + P_nt](t)t$$

Simplifying the equations above,

$$P_{0\ldots2}(t) = P_0(1-t)^2 + 2P_1(1-t)t + P_2t^2$$

$$P_{1\ldots 3}(t) = P_1(1-t)^2 + 2P_2(1-t)t + P_3t^2$$

$$\vdots$$

$$P_{n-2\ldots n}(t) = P_{n-2}(1-t)^2 + 2P_{n-1}(1-t)t + P_nt^2$$

These interpolation functions, again, can be used to perform another set of interpolation.

$$P_{0\ldots 3}(t) = P_{0\ldots 2}(1-t) + P_{1\ldots 3}t$$

$$P_{1\ldots 4}(t) = P_{1\ldots 3}(1-t) + P_{2\ldots 4}t$$

$$\vdots$$

$$P_{n-3\ldots n}(t) = P_{n-3\ldots n-1}(1-t) + P_{n-2\ldots n}t$$

Substituting the interpolation functions in the right-hand side with their function definitions simplifies to be the following:

$$P_{0\ldots 3}(t) = P_0(1-t)^3 + 3P_1(1-t)^2t + 3P_2(1-t)t^2 + P_3t^3$$

$$P_{1\ldots 4}(t) = P_1(1-t)^3 + 3P_2(1-t)^2t + 3P_3(1-t)t^2 + P_4t^3$$

$$\vdots$$

$$P_{n-3...n}(t) = P_{n-3}(1-t)^3 + 3P_{n-2}(1-t)^2t + 3P_{n-1}(1-t)t^2 + P_nt^3$$

Note how the coefficients of the terms seems to follow their corresponding binomial coefficient. However, until a proof is given for this conjecture, we shall leave them as some coefficients $a_0, a_1...a_n$. As such, continuing to repeat the algorithm would compute to be

$$P_{0...n}(t) = a_0P_0(1-t)^n + ... + a_kP_0(1-t)^{n-k}t^k + ... + a_nP_0t^n. \tag{8}$$

Eq. (8) can be rewritten as the following form:

$$P_{0...n}(t) = \sum_{i=0}^{n} a_iP_i(1-t)^{n-i}t^i \tag{9}$$

However, this is only an explicit definition of the interpolated point from a set of points, and we still do not have a single formula that recursively defines the De Casteljau algorithm itself. However, instead of strictly following the verbally-expressed algorithm, we can simply reverse it to express the construction in terms of two consecutive subset of control points or,

$$P_{a...b}(t) = [P_{a...b-1}(t)](1-t) + [P_{a+1...b}(t)]t, \tag{10}$$

where $P_{\nu\ldots\nu}(t) = P_{\nu}(t) = P_{\nu}, \nu \in \mathbb{Z}_{\geq 0}$.

However, it is still not yet found what the coefficients a_0, a_1 ... a_n are. An aforementioned remark was made upon how these coefficients seem to follow the binomial coefficients. As such, the following theorem can be made:

Theorem 1. *(Prautzsch et al., 2013) The coefficients of the terms of curve interpolation equal to the corresponding binomial coefficients;* $a_k = \binom{n}{k}, k \in [0, n] \cup \mathbb{Z}_{\geq 0}$

Proof. Let $\mathbf{V}$ be an arbitrary 2D vector. As such, translating the curve by $\mathbf{V}$ would be equivalent to adding $\mathbf{V}$ to every control point. Then,

$$\begin{aligned} P_{0\ldots n}(t) + \mathbf{V} &= \sum_{i=0}^{n} (P_i + \mathbf{V}) a_i (1-t)^{n-i} t^i \\ &= \sum_{i=0}^{n} P_i a_i (1-t)^{n-i} t^i + \mathbf{V} \sum_{i=0}^{n} a_i (1-t)^{n-i} t^i \\ &= P_{0\ldots n}(t) + \mathbf{V} \sum_{i=0}^{n} a_i (1-t)^{n-i} t^i \end{aligned} \tag{11}$$

In order for Eq. (11) to be true,

$$\sum_{i=0}^{n} a_i (1-t)^{n-i} t^i = 1. \tag{12}$$

Consider the following binomial expansion, as given by Prautzsch et al. (2013):

$$1 = ((1-t)+t)^n$$

By the binomial theorem, computing this binomial expansion gives,

$$((1-t)+t)^n = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i = 1. \tag{13}$$

with $\binom{n}{i}$ as the binomial coefficient, where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}.$$

Note how Eq. (13) is in equivalent to Eq. (12) with $a_i = \binom{n}{i}$. Thus, if $a_i = \binom{n}{i}$,

$$P_{0\ldots n}(t) + \mathbf{V} = P_{0\ldots n}(t) + \mathbf{V} \sum_{i=0}^{n} a_i (1-t)^{n-i} t^i.$$

is true. Therefore,

$$P_{0\ldots n}(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i. \tag{14}$$

In fact, this is the definition of the *Bézier curve*, and with the thorough knowledge of linear interpolation and the De Casteljau construction, we can now investigate and evaluate its properties.

4 Bézier Curve

In section 3, we discussed how repeated linear interpolation can be used to produce a smooth curve, which is formalized by the De Casteljau algorithm. In fact, the curve produced from this interpolation technique is called the **Bézier curve**.

4.1 Mathematical Definitions and Properties

To reiterate the aforementioned findings, the Bézier curve can be defined as an interpolation of a curve with a given set of control points. Mathematically, the Bézier curve $b(t)$ is defined as

$$b(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i, \tag{15}$$

where n is the degree of the curve. A degree of a curve is with respect to t. This simply means that, if the above definition were to be computed and simplified, it will yield a function of degree n with respect to t.

The polynomial expressed in the form of a sum in Eq. (13) is called the Bernstein polynomial $B^n(t)$, where n is the degree of the polynomials (Prautzsch et al., 2013).

$$B^n(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i. \tag{16}$$

Each individual term of this polynomial can be represented as $B_i^n(t)$. In other words,

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i. \tag{17}$$

Thus, the Bézier curve $b(t)$ can be simplified as the following:

$$b(t) = \sum_{i=0}^{n} P_i B_i^n(t), \tag{18}$$

When graphically plotted on a Cartesian plane, each term of Bernstein polynomial is an nth degree polynomial with respect to t, as shown in Fig. 4.1 with terms of Bernstein polynomial of degree 3. Note how at any t, the terms sum up to be 1, as deduced in Eq. (13).

The Bézier control points, then, can be thought of as adjusting the *weights* of these Bernstein polynomials. Such is why Bézier representation of a curve is referred to as the "weighted sum" (Prautzsch et al., 2013) of Bernstein polynomials.

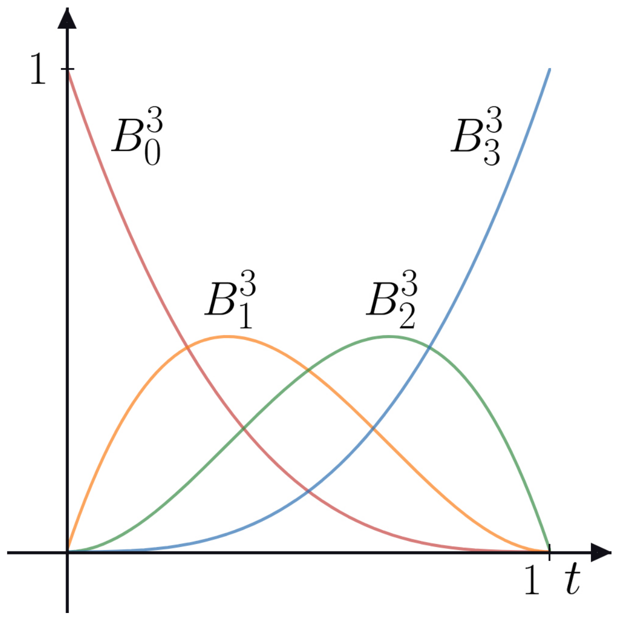

Figure 4.1: Terms of Cubic Bernstein Polynomials

4.1.1 End Points

Fig. 4.1 illustrates how at the end point of the curve $(t = 0, 1)$ the curve will always coincide with its initial and final Bézier control point, as shown in Fig. 4.2.

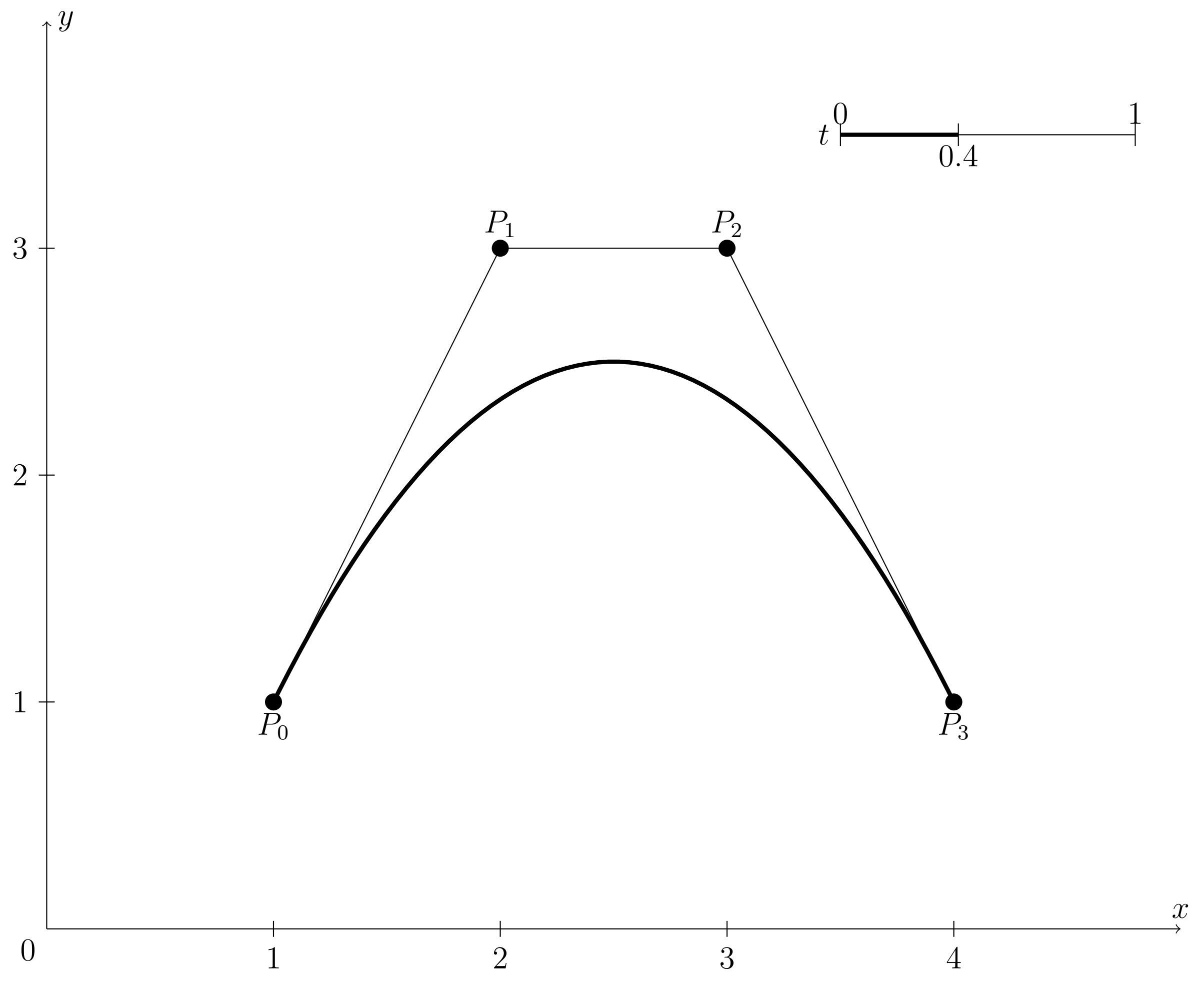

Figure 4.2: Illustration of the interpolated curve with 4 control points

As such, as noted by Prautzsch et al. (2013), for any Bézier curve, defined for $t \in [0, 1]$

$$b(0) = P_0, \tag{19}$$

$$b(1) = P_n. \tag{20}$$

4.1.2 Symmetry

Note how the graph illustrated in Fig. 4.1 is symmetric for line $t = 0.5$. This symmetry can be understood as the graph being identical to its original even after a horizontal translation of 1 towards left, and a horizontal reflection. As such, for any Bernstein polynomial

$$B_i^n(t) = B_{n-i}^n(1-t). \tag{21}$$

This symmetry, as noted by Prautzsch et al. (2013), implies that the Bézier curve

$$b(t) = \sum_{i=0}^{n} P_i B_i^n(t) = \sum_{i=0}^{n} P_i B_{n-i}^n(t) = \sum_{i=0}^{n} P_{n-i} B_i^n(t). \tag{22}$$

4.1.3 Recursion Formula

As noted by Prautzsch et al. (2013), the De Casteljau algorithm implies that the Bézier curve with control points $P_a, P_a + 1, ..., P_b - 1, P_b$

$$b(t) = \sum_{i=a}^{b} P_i B_i^n(t) = (1-t) \sum_{i=a}^{b-1} P_i B_i^{n-1}(t) + t \sum_{i=a+1}^{b} P_i B_i^{n-1}(t). \tag{23}$$

where n is the degree of the Bézier curve, which is equal to $b - a$.

4.2 Mathematical Advantage of Bézier Curves

- Ease of degree reduction: De Casteljau
- Ease of degree elevation (Exchange between legacy and new system)
- Unaffected by Translation and Rotation

4.3 Practical Advantage of Bézier Curves

- Intuitive to make tangentially continuous curves
- Ease of implementation

References

Bezier curves. (2017, December). Retrieved from https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Maya-Modeling/files/GUID-5A399D05-9271-4517-8370-0533DC468277-htm.html.

Bezier Pen Tool - Flash Professional CS5. (2017, April). Retrieved from https://helpx.adobe.com/flash/kb/bezier-pen-tool-flash-professional.html.

bubba (https://math.stackexchange.com/users/31744/bubba). (2015, December). *Bezier curve coefficients intuition.* Mathematics Stack Exchange. Retrieved from https://math.stackexchange.com/q/1589350.

de Casteljau, P. (1959). Outillages méthodes calcul. *André Citroën Automobiles SA, Paris*.

Prautzsch, H., Boehm, W., & Paluszny, M. (2013). *Bézier and B-spline techniques.* Springer Science & Business Media.

(bubba (https://math.stackexchange.com/users/31744/bubba), 2015)