Investigation on mathematical method used to construct curves in computer systems

Daekun Kim

May 6, 2018

Word count: 1447

Contents

1 Introduction

In the modern era, the computers are one of the most influential tools in humanity's lives. In fact, generations born after the 2000s, including myself, hardly lived an era with absence of heavy influence of computers. Many of its powerful abilities, such as being able to render high quality realistic images, is taken for granted by many people. It is capable of displaying millions of polygons and curved shapes in few milliseconds, which is crucial for many industrial applications, 3D video games, and for designers. However, computers, by nature, can only process discrete information, and it seems as if it must be only capable of rendering straight and rigid shapes; it is not clear how it is possible for them to portray such smooth curved surfaces and shapes in such a swift manner.

This rendering of curves and curved surfaces can be achieved using many methods, but one of the most prominent method is the Bézier curve. It is utilized in many of the famous 3D design and animation softwares, such as Adobe Flash (Chun, 2012), GIMP (Goelker, 2007), and Adobe Photoshop (Ulrich-Fuller & Fuller, 2007). Along with being one of the most popular methods to render curves in computer systems, the Bézier curves are also relatively intuitive and concise that it can be more easily understood than other more complicated methods, which will be explained throughout this essay.

This essay aims to **investigate the mathematical development of the Bézier curve, a method used to construct curves in computer systems**. I intend to connect the idea of rudimentary linear function to linear interpolation, and its parametric definition. Then, a natural realization of the De Casteljau algorithm, a recursive process used to construct the Bézier curve, from a repeated linear interpolation of attempting to draw a curve will be explained. Finally, the definition and the properties of Bézier curve itself will be discussed, with thorough examples of its real-world applications. Ultimately, the objective is to examine the development of the Bézier curve using the mathematical concepts taught in IB Mathematics HL.

2 Linear Interpolation

Straight lines are one of the easiest shapes one can imagine and draw, because of their simplicity to understand and manipulate. Such holds true even for computers. Graphically, drawing straight lines is the most rudimentary job a computer can perform. However, because it is so easy to render, it forms the fundamentals of how computers can render sophisticated curves and surfaces onto a screen.

Lines can be thought of as the shortest path that connects two points. A **linear interpolation**, then, is calculating which position on the line one would arrive on if one was to travel certain distance or time. Mathematically, given two points, (x_A, y_A) and (x_B, y_B) with $x_A < x_B$, a linear interpolation is construction of a new set of y from x values within $[x_A, x_B]$. The formula of linear interpolation can be given as a relationship of the slopes of the lines connecting the starting and end point with that of line connecting the interpolated point with the starting point, or

$$\frac{y - y_A}{x - x_A} = \frac{y_B - y_A}{x_B - x_A} \tag{1}$$

Eq. (1) is stating that, if a person was to travel over a line, the ratio of the distances at which one travelled vertically over the horizontal distance travelled must be the same no matter the position he or she lies on the line.

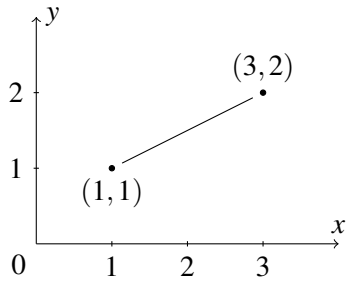

Figure 2.1: Linear interpolation between two points

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

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

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

The line shown in Fig 2.1 is called the **interpolant**, which can be thought of, literally, as a shortest path between two points. Solving Eq. (2) for x gives us the following:

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

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 nothing more than a linear function with a restricted domain between two points.

3 Parametric Equations

The previous section showed that linear interpolation is essentially plotting a point on a line using a linear function. However, such is not possible with our original definition of linear interpolation when the line cannot be defined using a function. For instance, consider a straight vertical line as shown in Fig 3.1.

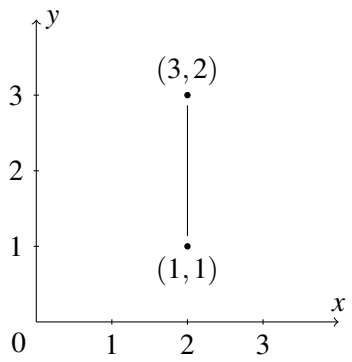

Figure 3.1: A vertical linear interpolation

In order for a line to be defined by a function, it must pass the Vertical Line Test, which states that there can only be one and only one y value for each x value. The line shown in Fig 3.1 obviously does not pass this condition, since there are infinitely many y values for $x = 2$.

However, by redefining linear interpolation with a set of parametric equations, one can easily overcome this apparent problem. Instead of defining an interpolation with respect to x, we can do so with another variable t within $[0,1]$, and define different equations for the values of y and x with no explicitly direct relationship between them. This new variable t can be thought of as the time, or the ratio of how much of the path one have travelled from the initial point to the total distance.

Example 3.1. Fig. 3.2 shows a linear interpolation at $t = 0.5$. At $t = 0.5$, the resulting point at halfway between P_A and P_B in both directions of the x- and y-axes. This can be thought of as a driver having travelled 50% of the path.

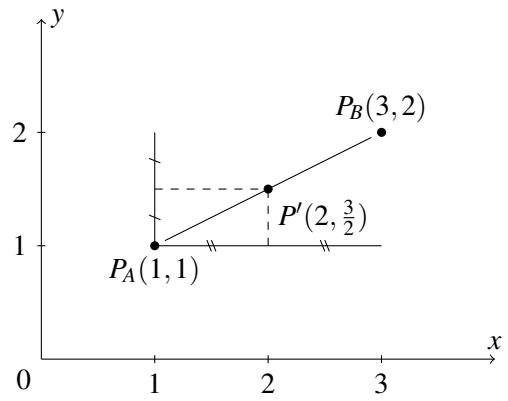

Figure 3.2: Parametric linear interpolation for $t = 0.5$

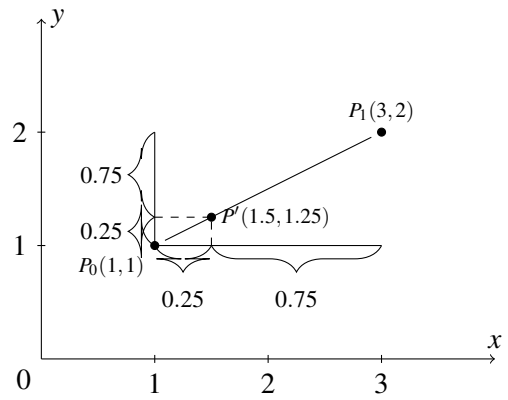

Figure 3.3: Parametric linear interpolation for $t = 0.25$

Example 3.2. In the same way, as shown in Fig. 3.3, at $t = 0.25$, the resulting points are one-quarter of the way between the initial and the terminal points in both axes, which is like travelling 25% of the path since the starting point.

As shown in the two examples above, the t value is analogous to the ratio in which the person has travelled from the initial point that a GPS device would let one know.

As such, a linear interpolation between (x_A, y_B) and (x_A, y_B) can be parametrically defined as following:

$$x(t) = x_A + (x_B - x_A)t \tag{4}$$

$$y(t) = y_A + (y_B - y_A)t \tag{5}$$

At $t = 0$, the interpolated point is at (x_A, y_A), the initial point, and at $t = 0$, it is at (x_1, y_1), the terminal point.

This can be considered as being at the initial point when one has travelled 0% of the path, and as being at the terminal point when one has travelled 100% of the path.

Parametric equations (4) and (5), however, look to be very similar. In fact, they can be combined to form a 2-dimensional (2D) vector equation. Let P_0 and P_1 be position vectors, where $P_0 = (x_0, y_0)$ and $P_1 = (x_1, y_1)$. Then,

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

where $P(t)$ is a function that linearly interpolates a point between P_0 and P_1 for $t \in [0,1]$. Analogously, this is a vector addition between the initial location and the path a driver has travelled so far at time t. 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" (Prautzsch, Boehm, & Paluszny, 2013) of the resulting point P with respect to the initial and final points. More specifically, the ratio $t : (1-t)$ equals to the ratio of the distance from the current point P with the initial point P_0 and the final point P_1, respectively, as shown in Fig. 3.4.

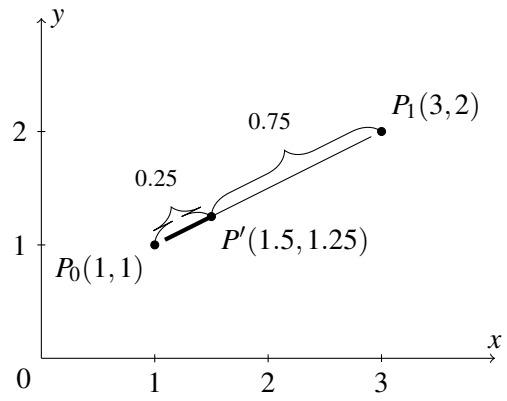

Figure 3.4: Ratio of $(1-t)$ and t for $t = 0.25$

To construct the interpolant, t values with an infinitesimal interval between each other would be used. In computer systems, this would translate to having t values with an interval small enough that will allow the

line to appear smooth and continuous, and large enough to not harm the performance of the computer.

4 Recursive Linear Interpolation

Thus far, the main focus of our investigation was on the methods used to produce straight lines. However, surprisingly, linear interpolation can be repeated to produce a smooth curve. The following example will show how such can be achieved with 3 points.

Example 4.1. Let $P_0 = (1,1)$, $P_1 = (2,2)$, and $P_2 = (3,1)$. By applying linear interpolation, we know that we can produce straight lines between these three points using the vector equation $P(t) = P_A(1-t) + P_B t$. This is shown in Fig. 4.1.

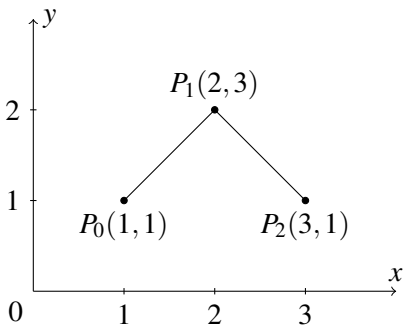

Figure 4.1: Linear interpolants between P_0 and P_1, and P_1 and P_2

Let $P_{0\ldots1}$ be a linearly interpolated point between P_0 and P_1 at $t = 0.5$. Similarly, let $P_{1\ldots2}$ be the linearly interpolated point between P_1 and P_2 at $t = 0.5$. These newly interpolated points can be used as the starting and terminal points of new linear interpolation, as shown in Fig. 4.2.

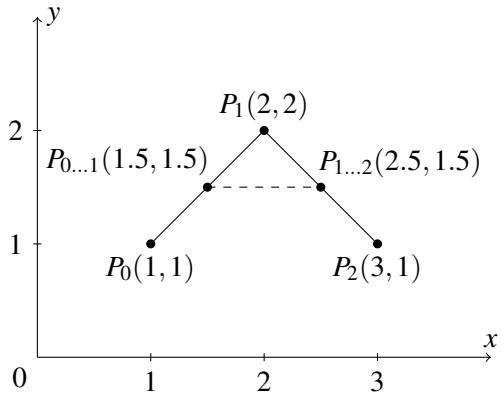

Figure 4.2: Linear interpolant of $P_{0\ldots1}$ and $P_{1\ldots2}$

Finally, let $P_{0...2}$ be the linearly interpolated point between $P_{0...1}$ and $P_{1...2}$. $P_{0...2}$, then, is a resulting point of the quadratic interpolation from points P_0, P_1, and P_2 at $t = 0.5$, as shown in Fig. 4.3.

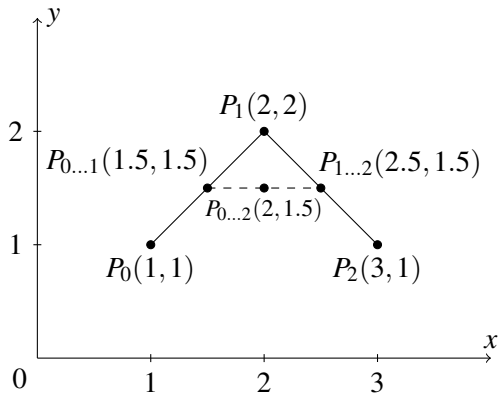

Figure 4.3: Quadratic interpolation at $t = 0.5$

This technique of repeated linear interpolation can be performed at $t = 0.25$ and $t = 0.75$, as shown in Fig. 4.4 and 4.5.

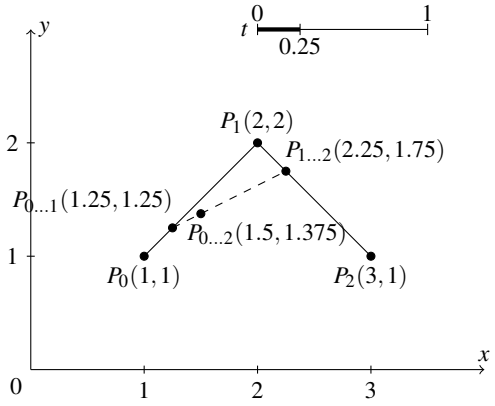

Figure 4.4: Quadratic interpolation at $t = 0.25$

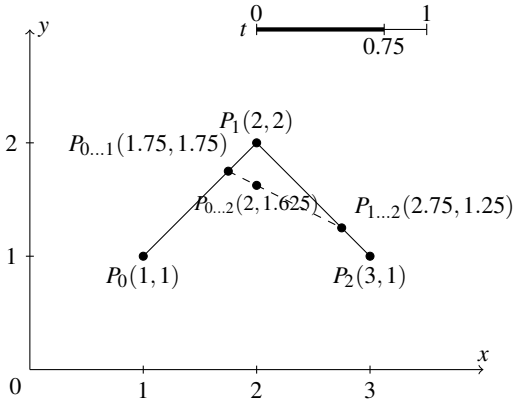

Figure 4.5: Quadratic interpolation at $t = 0.75$

At $t = 0$ and $t = 1$, the quadratic interpolation always yields points identical to the first and last of the original point, respectively.

Example 4.2. The points P_0, P_1 and P_2 are defined as that of example 4.1. At $t = 0$, the two linearly interpolated points are

$$
\begin{aligned}
P_{0\ldots1}(0) &= P_0(1-0)+P_1(0) \\
&= P_0. \\
P_{1\ldots2}(0) &= P_1(1-0)+P_2(0) \\
&= P_1.
\end{aligned}
$$

Then, the resulting point from the quadratic interpolation is

$$
\begin{aligned}
P_{0\ldots2}(0) &= P_{0\ldots1}(1-0)+P_{1\ldots2}(0) \\
&= P_{0\ldots1} \\
&= P_0,
\end{aligned}
$$

which is the first of the original points, P_0.

Example 4.3. Given the same three points as Examples 4.1 and 4.2, at $t = 1$, the two linearly interpolated points are

$$
\begin{aligned}
P_{0\ldots1}(1) &= P_0(1-1)+P_1(1) \\
&= P_1. \\
P_{1\ldots2}(1) &= P_1(1-1)+P_2(1) \\
&= P_2.
\end{aligned}
$$

Then, the resulting point from the quadratic interpolation is

$$
\begin{aligned}
P_{0\ldots2}(1) &= P_{0\ldots1}(1-1)+P_{1\ldots2}(1) \\
&= P_{1\ldots2} \\
&= P_2,
\end{aligned}
$$

which is the last of the original points, P_2.

With infinitesimal interval of t values, the quadratic interpolant can be consturcted, as shown in Fig. 4.6. In computer systems, this would translate to having small enough interval of t value that will allow the curve to look smooth, and also one that is large enough that will not harm the computer's performance with redundant calculations.

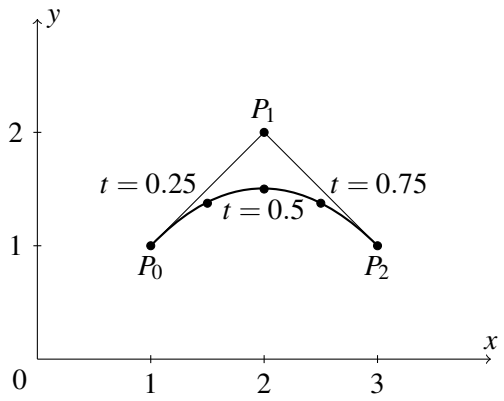

Figure 4.6: Quadratic interpolant

This repeated linear interpolation is called the **De Casteljau construction**, or the **De Casteljau algorithm** (de Casteljau, 1959). Because this algorithm repeats itself by producing new interpolation from previously interpolated points, it is described as being "recursive" (Prautzsch et al., 2013), meaning that it repeats itself to meet the objective. The curve produced from this construction is later formalized and popularized by Pierre Bézier (Farin, 2001), by whom this curve, the **Bézier curve**, was named after.

In the De Casteljau algorithm, the only information needed to construct any curve is the initial points, such as P_0, P_1 and P_2 shown in Fig. 4.6. As such, they are the ones that controls the appearance of the resulting curve. Hence, these points are referred to as the **control points** of the Bézier curve (Prautzsch et al., 2013).

Thus far, we have investigated on construction of quadratic curves with three Bézier control points. However, by following the similar process of repeating linear interpolation, curves with higher number of control points can be easily constructed, as shown in the following example.

Example 4.4. Let $P_0 = (1,1), P_1 = (1,5), P_2 = (5,5)$, and $P_3 = (5,1)$. With linear interpolation performed for each consecutive pairs of control points, a set of straight interpolants can be produced as shown in Fig. 4.7.

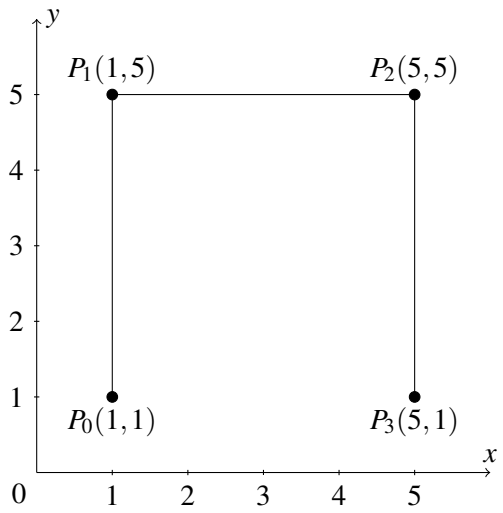

Figure 4.7: Linear interpolants between the four Bézier control points

In order to produce an interpolated point using the control points, the points $P_{0...1}, P_{1...2}$ and $P_{2...3}$ must be interpolated first. These points can be used to perform another set of linear interpolations, as shown in Fig. 4.8.

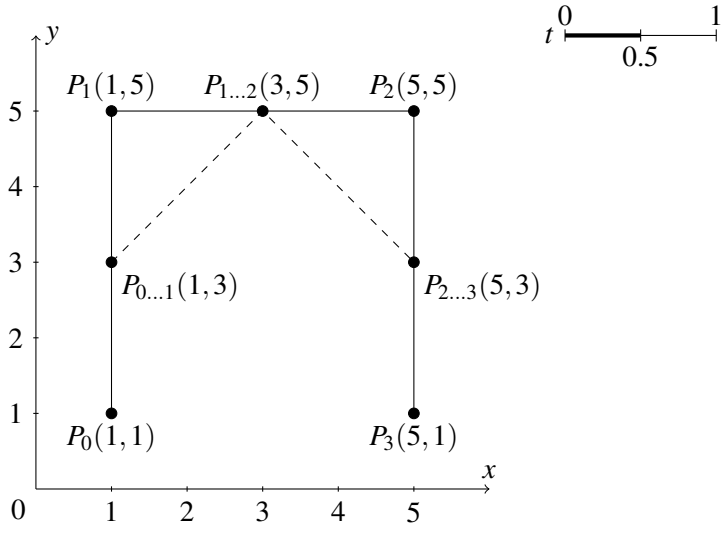

Figure 4.8: Second recursion of De Casteljau algorithm

Then, the points $P_{0...1}, P_{1...2}$ and $P_{2...3}$ can be used to perform another linear interpolation, as shown in Fig. 4.9.

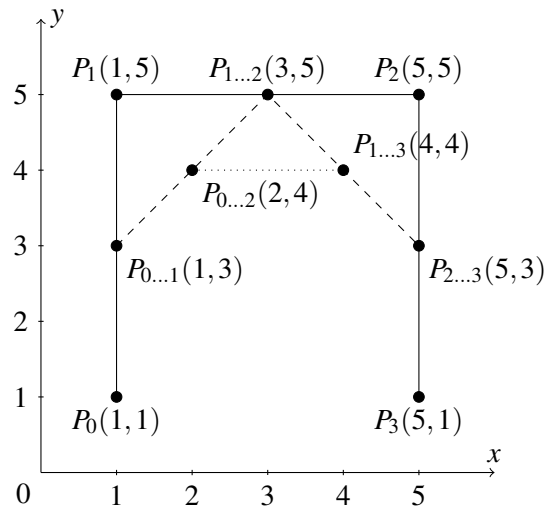

Figure 4.9: Third recursion of De Casteljau algorithm

Finally, the resulting point, $P_{0...3}$, of the cubic interpolation at $t = 0.5$ is obtained using the points $P_{0...2}$ and $P_{1...3}$, as shown in Fig. 4.10:

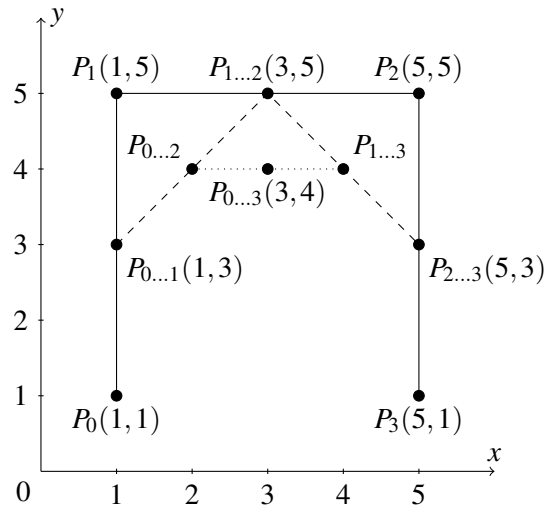

Figure 4.10: Resulting point of the cubic interpolation

As such, the De Casteljau algorithm can be expressed as the following set of procedure:

1. Interpolate a set of points at $t \in [0,1]$ for each of the consecutive pair of control points.
2. Repeat step #1 with newly interpolated points, until a single point is left.

Thus far, the Bézier curve has vaguely been mentioned, and its mathematical definition is yet unclear.

However, this algorithm can be used to deduce the formal definition of a Bézier curve. 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 $a \geq b$. If $a = b$, $P_{a\ldots b}(t) = P_a = P_b$. Next, let the control points of a Bézier curve be $P_0, P_1, P_2, \ldots, P_{n-1}, P_n$. Performing a set of linear interpolation to consecutive pairs of control points at t gives the following:

$$\begin{aligned}
P_{0\ldots1}(t) &= P_0(1-t) + P_1 t, \\
P_{1\ldots2}(t) &= P_1(1-t) + P_2 t, \\
P_{2\ldots3}(t) &= P_2(1-t) + P_3 t, \\
&\vdots \\
P_{n-1\ldots n}(t) &= P_{n-1}(1-t) + P_n t.
\end{aligned}$$

The newly interpolated points can than be used to perform another set of interpolation.

$$\begin{aligned}
P_{0\ldots2}(t) &= [P_{0\ldots1}(t)](1-t) + [P_{1\ldots2}(t)]t, \\
P_{1\ldots3}(t) &= [P_{1\ldots2}(t)](1-t) + [P_{2\ldots3}(t)]t, \\
P_{2\ldots4}(t) &= [P_{2\ldots3}(t)](1-t) + [P_{3\ldots4}(t)]t, \\
&\vdots \\
P_{n-2\ldots n}(t) &= [P_{n-2\ldots n-1}(t)](1-t) + [P_{n-1\ldots n}(t)]t.
\end{aligned}$$

Substituting the interpolation functions with their definitions,

$$\begin{aligned}
P_{0\ldots2}(t) &= [P_0(1-t) + P_1 t](1-t) + [P_1(1-t) + P_2 t]t, \\
P_{1\ldots3}(t) &= [P_1(1-t) + P_2 t](1-t) + [P_2(1-t) + P_3 t]t, \\
P_{2\ldots4}(t) &= [P_2(1-t) + P_3 t](1-t) + [P_3(1-t) + P_4 t]t, \\
&\vdots \\
P_{n-2\ldots n}(t) &= [P_{n-2}(1-t) + P_{n-1} t](1-t) + [P_{n-1}(1-t) + P_n t]t.
\end{aligned}$$

Simplifying the equations above,

$$\begin{aligned}
P_{0\ldots2}(t) &= P_0(1-t)^2 + 2P_1(1-t)t + P_2 t^2, \\
P_{1\ldots3}(t) &= P_1(1-t)^2 + 2P_2(1-t)t + P_3 t^2, \\
P_{2\ldots4}(t) &= P_2(1-t)^2 + 2P_3(1-t)t + P_4 t^2, \\
&\vdots \\
P_{n-2\ldots n}(t) &= P_{n-2}(1-t)^2 + 2P_{n-1}(1-t)t + P_n t^2.
\end{aligned}$$

Following the same procedure of interpolating and simplifying yields the following:

$$\begin{aligned}
P_{0\ldots3}(t) &= P_0(1-t)^3 + 3P_1(1-t)^2t + 3P_2(1-t)t^2 + P_3t^3,\\
P_{1\ldots4}(t) &= P_1(1-t)^3 + 3P_2(1-t)^2t + 3P_3(1-t)t^2 + P_4t^3,\\
P_{2\ldots5}(t) &= P_2(1-t)^3 + 3P_3(1-t)^2t + 3P_4(1-t)t^2 + P_5t^3,\\
&\vdots\\
P_{n-3\ldots n}(t) &= P_{n-3}(1-t)^3 + 3P_{n-2}(1-t)^2t + 3P_{n-1}(1-t)t^2 + P_nt^3.
\end{aligned}$$

Repeating the same procedure again,

$$\begin{aligned}
P_{0\ldots4}(t) &= P_0(1-t)^4 + 4P_1(1-t)^3t + 6P_2(1-t)^2t^2 + 4P_3(1-t)t^3 + P_4t^4,\\
P_{1\ldots5}(t) &= P_1(1-t)^4 + 4P_2(1-t)^3t + 6P_3(1-t)^2t^2 + 4P_4(1-t)t^3 + P_5t^4,\\
P_{2\ldots6}(t) &= P_2(1-t)^4 + 4P_3(1-t)^3t + 6P_4(1-t)^2t^2 + 4P_5(1-t)t^3 + P_6t^4,\\
&\vdots\\
P_{n-4\ldots n}(t) &= P_{n-4}(1-t)^4 + 4P_{n-3}(1-t)^3t + 6P_{n-2}(1-t)^2t^2 + 4P_{n-1}(1-t)t^3 + P_nt^4.
\end{aligned}$$

A pattern that imitates binomial expansion of $((1-t)+t)^n$ emerges, along with the control points multiplied as coefficients of each terms. Continuing on a similar manner, the complete equation of interpolated Bézier curve is obtained.

$$\begin{aligned}
P_{0\ldots n}(t) = P_0\binom{n}{0}(1-t)^n + P_1\binom{n}{1}(1-t)^{n-1}t + P_2\binom{n}{2}(1-t)^{n-2}t^2\\
+\cdots+P_k\binom{n}{k}(1-t)^{n-k}t^k+\cdots+P_{n-1}\binom{n}{n-1}(1-t)t^{n-1}+P_n\binom{n}{n}t^n, \quad (8)
\end{aligned}$$

where $\binom{n}{k}$ is a binomial coefficient ($\binom{n}{k} = \frac{n!}{k!(n-k)!}$). Rewriting the equation above,

$$P_{0\ldots n}(t) = \sum_{i=0}^{n} P_i\binom{n}{i}(1-t)^{n-1}t^i. \quad (9)$$

This, in fact, is the definition of the Bézier curve, which is written as $b_n(t)$ for curves with $n+1$ control points. As such, the mathematical development of Bézier curves from simple linear interpolation has been investigated so far. The following part of the essay will examine the properties of Béizer curves and why they allow is to be one of the most favoured method of describing curves in computer systems.

References

Chun, R. (2012). *Adobe Flash Professional CS6: Classroom in a book.* Adobe Press.

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

Farin, G. (2001). *Curves and surfaces for CAGD: A practical guide* (5th Edition ed.). Elsevier Science.

Goelker, K. (2007). *Gimp 2 for photographers: Image editing with open source software*. Rocky Nook.

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

Ulrich-Fuller, L., & Fuller, R. (2007). *Photoshop CS3 Bible*. John Wiley & Sons.