How can the Bézier curve be used to approximate a cycloid?

Daekun Kim

May 19, 2018

Word count: 5161

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.

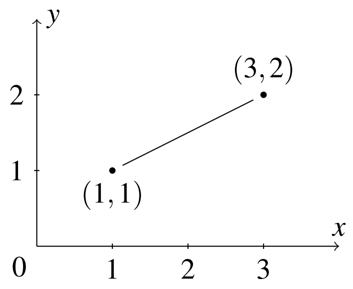

Figure 2.1: Linear interpolation between two points

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.

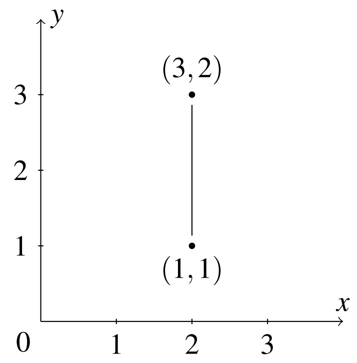

Figure 3.1: A vertical linear interpolation

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.

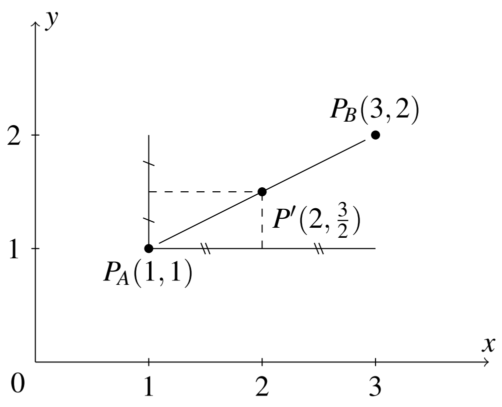

Figure 3.2: Parametric linear interpolation for $t = 0.5$

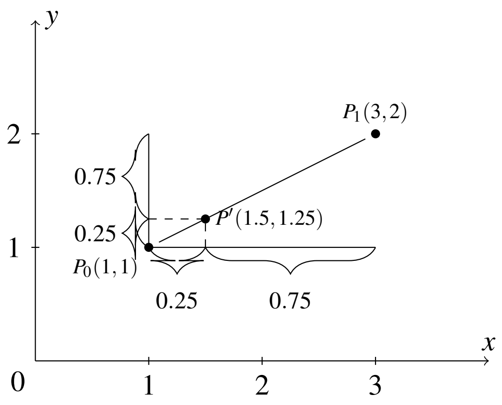

Figure 3.3: Parametric linear interpolation for $t = 0.25$

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.

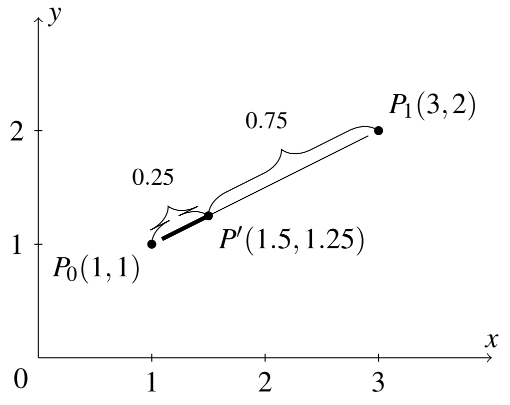

Figure 3.4: Ratio of $(1-t)$ and t for $t = 0.25$

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.

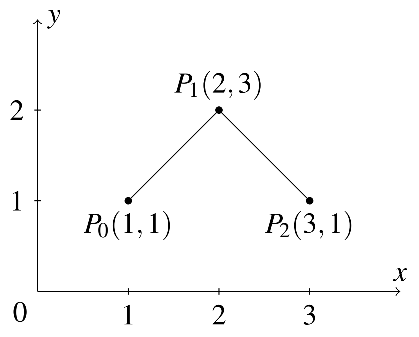

Figure 4.1: Linear interpolants between P_0 and P_1, and P_1 and P_2

Let $P_{0...1}$ be a 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 $t = 0.5$. These newly interpolated points can be used as the starting and terminal points of new linear interpolation, as shown in Fig. 4.2.

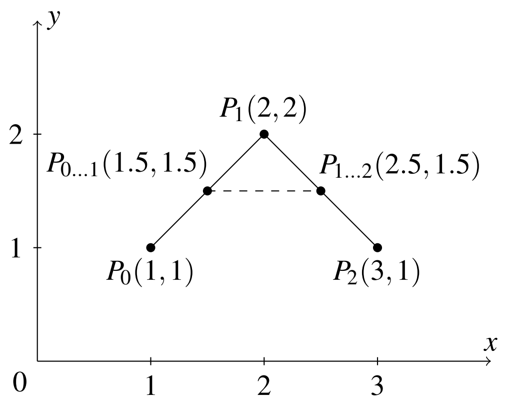

Figure 4.2: Linear interpolant of $P_{0...1}$ and $P_{1...2}$

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.

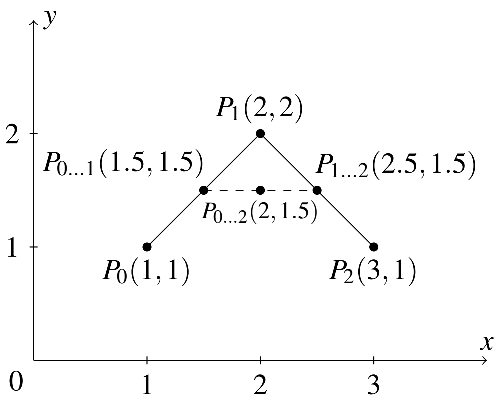

Figure 4.3: Quadratic interpolation at $t = 0.5$

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.

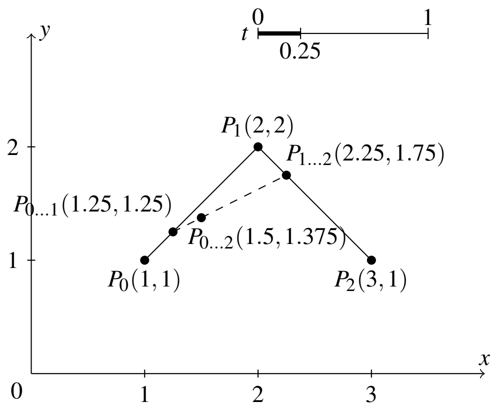

Figure 4.4: Quadratic interpolation at $t = 0.25$

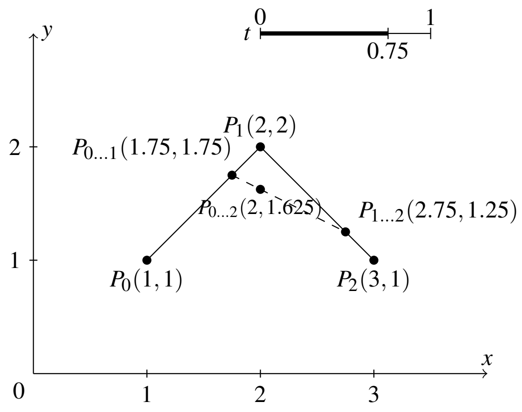

Figure 4.5: Quadratic interpolation at $t = 0.75$

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.

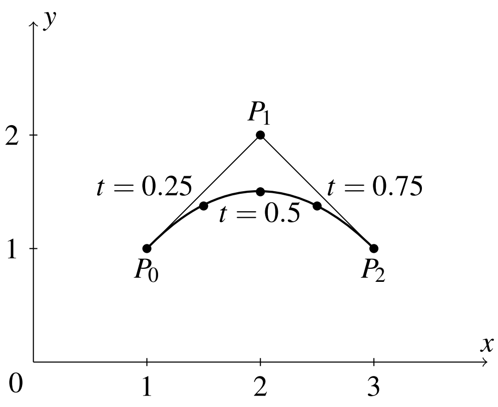

Figure 4.6: Quadratic interpolant

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.

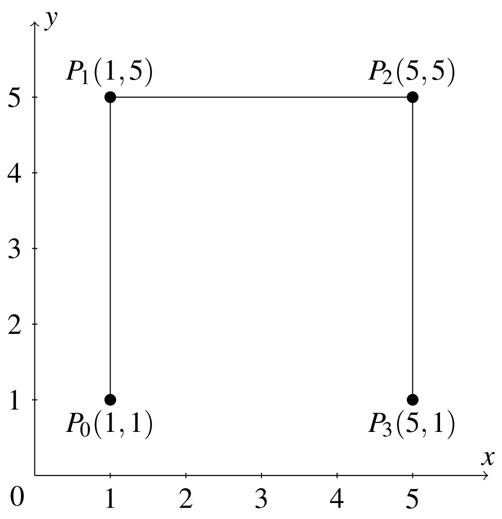

Figure 4.7: Linear interpolants between the four Bézier control points

In order to produce an interpolated point using the control points, the points $P_{0\ldots1}, P_{1\ldots2}$ and $P_{2\ldots3}$ must be interpolated first. These points can be used to perform another set of linear interpolations, as shown in Fig. 4.8.

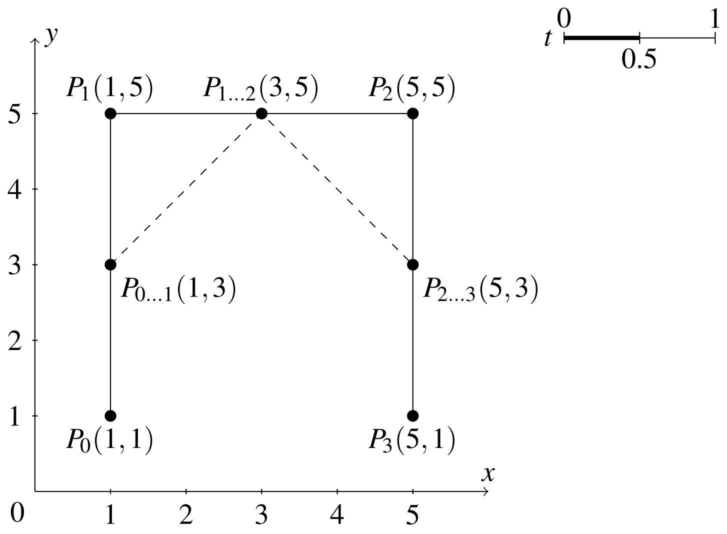

Figure 4.8: Second recursion of De Casteljau algorithm

Then, the points $P_{0\ldots1}, P_{1\ldots2}$ and $P_{2\ldots3}$ can be used to perform another linear interpolation, as shown in Fig. 4.9.

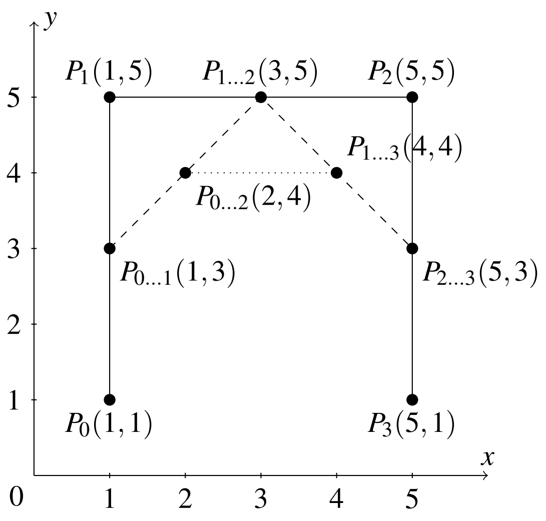

Figure 4.9: Third recursion of De Casteljau algorithm

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:

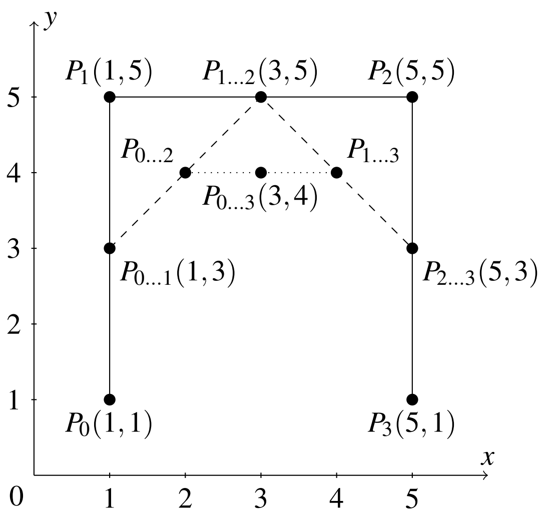

Figure 4.10: Resulting point of the cubic interpolation

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:

$$\begin{aligned} P_{0\ldots 1}(t) &= P_0(1-t) + P_1 t, \\ P_{1\ldots 2}(t) &= P_1(1-t) + P_2 t, \\ P_{2\ldots 3}(t) &= P_2(1-t) + P_3 t, \\ &\vdots \\ P_{n-1\ldots n}(t) &= P_{n-1}(1-t) + P_n t. \end{aligned}$$

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

$$\begin{aligned} P_{0\ldots 2}(t) &= [P_{0\ldots 1}(t)](1-t) + [P_{1\ldots 2}(t)]t, \\ P_{1\ldots 3}(t) &= [P_{1\ldots 2}(t)](1-t) + [P_{2\ldots 3}(t)]t, \\ P_{2\ldots 4}(t) &= [P_{2\ldots 3}(t)](1-t) + [P_{3\ldots 4}(t)]t, \\ &\vdots \\ P_{n-2\ldots n}(t) &= [P_{n-2\ldots n-1}(t)](1-t) + [P_{n-1\ldots n}(t)]t. \end{aligned}$$

Substituting the interpolation functions with their definitions,

$$\begin{aligned} P_{0\ldots 2}(t) &= [P_0(1-t) + P_1 t](1-t) + [P_1(1-t) + P_2 t]t, \\ P_{1\ldots 3}(t) &= [P_1(1-t) + P_2 t](1-t) + [P_2(1-t) + P_3 t]t, \\ P_{2\ldots 4}(t) &= [P_2(1-t) + P_3 t](1-t) + [P_3(1-t) + P_4 t]t, \\ &\vdots \\ P_{n-2\ldots n}(t) &= [P_{n-2}(1-t) + P_{n-1} t](1-t) + [P_{n-1}(1-t) + P_n t]t. \end{aligned}$$

Simplifying the equations above,

$$\begin{aligned} P_{0\ldots 2}(t) &= P_0(1-t)^2 + 2P_1(1-t)t + P_2 t^2, \\ P_{1\ldots 3}(t) &= P_1(1-t)^2 + 2P_2(1-t)t + P_3 t^2, \\ P_{2\ldots 4}(t) &= P_2(1-t)^2 + 2P_3(1-t)t + P_4 t^2, \\ &\vdots \\ P_{n-2\ldots n}(t) &= P_{n-2}(1-t)^2 + 2P_{n-1}(1-t)t + P_n t^2. \end{aligned}$$

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,

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

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.

5 Definition and Important Properties of Bézier Curves

Thus far, we have investigated on the mathematical development of Bézier curves. However, before the attempt of approximating a cycloid can be made, the knowledge of definition and some important properties of Bézier curves must precede.

To reiterate the definition of Bézier curves,

Definition 1 (Prautzsch et al. (2013))**.**

$$b_n(t) = P_{0\ldots n} = \sum_{i=0}^{n} P_i \binom{n}{i} (1-t)^{n-1} t^i. \tag{10}$$

Although there are numerous properties of Bézier curves, the two most important ones are regarding their initial and terminal points, and the tangent at both points.

Property 1. At $t = 0$ and $t = 1$, the interpolated points coincide with the first and last Bézier control points, respectively (Prautzsch et al., 2013).

This will be useful when determining the location of the first an last control points of Bézier approximation based on the original curve.

Property 2. The tangents at $t = 0$ and $t = 1$ pass through points P_0, P_1 and P_{n-1}, P_n, respectively (Prautzsch et al., 2013).

This property is not so easy to realize without a visual intuition. As such, the next example graphically investigates this property to aid the understanding.

Example 5.1. Let $P_0 = (1,1), P_1 = (2,3), P_2 = (3,3)$, and $P_3 = (4,1)$. Plotting the control points and the Bézier curve yields the following:

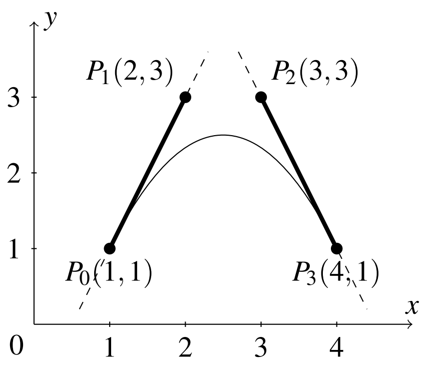

Figure 5.1: Tangent lines of a cubic Bézier curve

The tangents at the initial and terminal points ($t = 0$ and $t = 1$) of the curve show that they indeed pass through P_0, P_1 and P_{n-1}, P_n (P_2, P_3 in this case), respectively.

By exploting the properties of the Bézier curve, we can now make different approaches to approximating a cycloid using a Bézier curve.

6 Approximation of Cycloids

As mentioned in the introduction of this essay, the Bézier curve is one of the most predominant means of representing curves and shapes on computer systems. As such, the topic of representing various kinds of curves, such as circles and ellipses, has been thoroughly investigated along with the improvement of computer graphics. However, because the Bézier curve is essentially a parametric polynomial curve, it cannot give an exact, mathematically equivalent representation of non-polynomial curves, such as circles, ellipses, and graphs of transcendental functions, such as that of sine and cosine function. However, there has been many successful attempts in approximating different kinds of curves, especially regarding the conic sections, such as an approximation given by Dokken et al (1989), which approximates a unit circle with a maximum error in radius of $2 * 10^{-6}$ with only 8 piecewise Bézier curves. However, after a thorough research, it was evident that there is no formal attempt of approximation done for cycloids using a Bézier curve. This came to me as a shock, because cycloids themselves are defined parametrically like Bézier curves, and it is also relatively easy to understand and manipulate. As such, I have determined to investigate on the extent in which a Bézier curve can be used to approximate a cycloid.

Cycloid is a parametric curve that is defined as following:

$$\begin{aligned} x(\theta) &= \theta - \sin\theta, \\ y(\theta) &= 1 - \cos\theta. \end{aligned} \tag{11}$$

This is a curve that is generated by plotting the path in which a point on a circle that is rolling on a smooth, straight surface passes through. Graphically, it is illustrated by Fig. 6.1

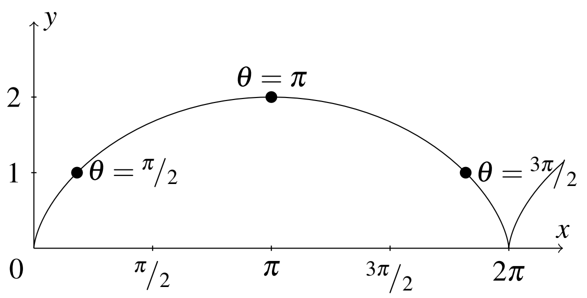

Figure 6.1: An arc of cycloid

Because this curve repeats infinitely, approximating one arc will yield a complete repeatable approximation of the curve. Moreover, since an arc of a cycloid is symmetrical at $\theta = \pi$, we only need to approximate $\theta \in [0, \pi]$ section of the arc, which is illustrated in Fig. 6.2.

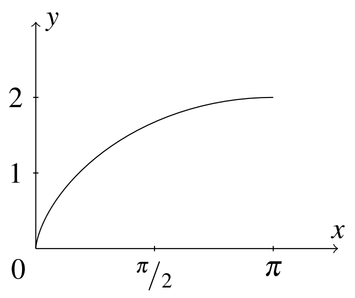

Figure 6.2: Section of cycloid for $\theta \in [0, \pi]$

In order to make the comparison between the cycloid and the Bézier approximation easier, the cycloid can be redefined as following with respect to variable t so that the arc in Fig 6.2 will be constructed for $t \in [0, 1]$.

$$\begin{aligned} x(t) &= \pi t - \sin \pi t, \\ y(t) &= 1 - \cos \pi t. \end{aligned} \tag{12}$$

The three methods of approximation that will be investigated in this paper will be based on the Maclaurin polynomial, approximation of unit circle given by Dokken, Daehlen, Lyche, and Morken (1990), and numerical analysis. Although these are not the only methods of approximating a non-polynomial curve with a Bézier curve, they will allow a through comparison and holistic judgement of Bézier curve as a means of approximation to be made.

In all three approximations, cubic Bézier curves will be used, because they are most commonly used in most of the popular computer-aided design (CAD) softwares, such as Photoshop (Ulrich-Fuller & Fuller, 2007) and GIMP (Goelker, 2007), portraying them as being much more practical than curves of other degrees.

When evaluating the approximation, two approaches may be used: Extent of tangential approximation at initial and terminal points, and numerical error analysis.

In Fig. 6.2, it shows that the initial and terminal points of the cycloid are $(0,0)$ and $(\pi,2)$, respectively. Moreover, the calculating the derivatives at $t=0$ and $t=1$ also shows that it has vertical and horizontal tangents at the initial and terminal points, respectively.

$$\begin{aligned}\frac{dy}{dx} &= \frac{\frac{d}{dt}(1-\cos(\pi t)}{\frac{d}{dt}(\pi t-\sin(\pi t)} \\ &= \frac{\pi\sin(\pi t)}{\pi-\pi\cos(\pi t)}.\end{aligned}$$

By L'Hôpital's rule of indeterminate forms, at the initial point $t=0$,

$$\begin{aligned}\frac{dy}{dx}|_{t=0} &= \lim_{t\to0^+}\frac{\pi\sin(\pi t)}{\pi-\pi\cos(\pi t)} \\ &= \lim_{t\to0^+}\frac{\pi^2\cos(\pi t)}{\pi^2\sin(\pi t)} \\ &= +\infty \text{ (Vertical tangent)}\end{aligned}$$

At the terminal point $t=1$,

$$\begin{aligned}\frac{dy}{dx}|_{t=1} &= \frac{\pi\sin(\pi*0)}{\pi-\pi\cos(\pi*0)} \\ &= 0 \text{ (Horizontal tangent).}\end{aligned}$$

As such, assessing if the approximated curve tangentially coincides with the original cycloid will allow us to assess the quality of the approximation to some extent.

For the numerical analysis, we must define a error function that will yield numerical values for the accuracies of different approximations, which will allow a reasoned comparison to be made. Using the Pythagoras theorem, the Euclidean distance between the cycloid and the Bézier approximation at $t\in[0,1]$ can be computed, as following:

$$\varepsilon(t) = \sqrt{(x(t) - b_x(t))^2 + (y(t) - b_y(t))^2}, \tag{13}$$

where $x(t)$ and $y(t)$ are the x and y components of the parametric equations of a cycloid, and $b_x(t)$ and x and y components of the Bézier approximation. Thus, taking $\max\limits_{t\in[0,1]} \varepsilon(t)$, we can evaluate the maximum error of the approximation, obtaining another way to assess the quality of the approximation to some extent.

6.1 Maclaurin Polynomial

A Maclaurin polynomial $P_n(x)$ approximates a function $f(n)$ with order n as following:

$$f(x) \approx P_n(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!} + \ldots + \frac{f^{(n)}(0)}{n!} \tag{14}$$

Using the Maclaurin polynomial, the parametric equations of cycloid can be approximated. Because a cubic Bézier curve is used, a Maclaurin polynomial with order of 3 must be used.

$$\begin{aligned}
x(t) &\approx x(0) + x'(0)t + \frac{x''(0)}{2!}t^2 + \frac{x'''(0)}{3!}t^3 \\
&= (\pi t - sin\pi t)|_{t=0} + (\pi - \pi cos\pi t)|_{t=0} + \frac{(\pi^2 sin\pi t)|_{t=0}}{2}t^2 + \frac{(\pi^3 cos\pi t)|_{t=0}}{6}t^3 \\
&= \frac{\pi^3}{6}t^3 \\
y(t) &\approx y(0) + y'(0)t + \frac{y''(0)}{2!}t^2 + \frac{y'''(0)}{3!}t^3 \\
&= (1 - cos\pi t)\_t = 0 + (\pi sin\pi t)|_{t=0}t + \frac{(\pi^2 cos\pi t)|_{t=0}}{2}t^2 + \frac{(-\pi^3 sin\pi t)|_{t=0}}{6}t^3 \\
&= \frac{\pi^2}{2}t^2
\end{aligned}$$

Let $b(t)$ be a cubic Bézier approximation of cycloid. To rearrange the Maclaurin polynomial into a Bézier form, we must first expand and simplify the definition of Bézier curve into a polynomial in terms of t.

$$\begin{aligned}
b(t) &= \sum_{i=0}^{3} P_i \binom{3}{i} (1-t)^{3-i} t^i \\
&= P_0(1-t)^3 + 3P_1(1-t)^2 t + 3P_2(1-t)t^2 + P_3 t^3 \\
&= P_0(1-3t+3t^2+t^3) + 3P_1(t-2t^2+t^3) + 3P_2(t^2-t^3) + P_3 t^3 \\
&= P_0 - 3P_0 t + 3P_0 t^2 - P_0 t^3 + 3P_1 t - 6P_1 t^2 + 3P_1 t^3 + 3P_2 t^2 - 3P_2 t^3 + P_3 t^3 \\
&= P_0 + 3(P_1 - P_0)t + 3(P_2 - 2P_1 + P_0)t^2 + (P_3 - 3P_2 + 3P_1 - P_0)t^3
\end{aligned}$$

Since

$$b_x(t) \approx x(t) \approx \frac{\pi^3}{6} t^3 \qquad \text{and} \qquad b_y(t) \approx y(t) \approx \frac{\pi^2}{2} t^2,$$

we can equate the coefficients of each terms of $b(t)$ with those of Maclaurin approximations of $x(t)$ and $y(t)$ to find the control points. Firstly, for the x component of the control points, we obtain 4 simultaneous equations as following:

$$\begin{aligned}
P_{0,x} &= 0, \\
3(P_{1,x} - P_{0,x}) &= 0, \\
3(P_{2,x} - 2P_{1,x} + P_{0,x}) &= 0, \\
P_{3,x} - 3P_{2,x} + 3P_{1,x} - P_{0,x} &= \frac{\pi^3}{6}.
\end{aligned}$$

Solving the simultaneous equations above yields

$$\begin{aligned}
\therefore P_{0,x} &= 0, \\
\therefore P_{1,x} &= 0, \\
\therefore P_{2,x} &= 0, \\
\therefore P_{3,x} &= \frac{\pi^3}{6}.
\end{aligned}$$

Then, for the y component of the control points, we have another 4 simultaneous equations as following:

$$\begin{aligned}
P_{0,y} &= 0, \\
3(P_{1,y} - P_{0,y}) &= 0, \\
3(P_{2,y} - 2P_{1,y} + P_{0,y}) &= \frac{\pi^2}{2}, \\
P_{3,y} - 3P_{2,y} + 3P_{1,y} - P_{0,y} &= 0.
\end{aligned}$$

Solving the simultaneous equations above yields

$$\therefore P_{0,y} = 0,$$
$$\therefore P_{1,y} = 0,$$
$$\therefore P_{2,y} = \frac{\pi^2}{6},$$
$$\therefore P_{3,y} = \frac{\pi^2}{2}.$$

As such, the cubic Maclaurin approximation of cycloid yields the following Bézier control points:

$$P_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$P_2 = \begin{bmatrix} 0 \\ \frac{\pi^2}{6} \end{bmatrix}, \qquad P_3 = \begin{bmatrix} \frac{\pi^3}{6} \\ \frac{\pi^2}{2} \end{bmatrix}$$

This approximation is shown in Fig. 6.6.

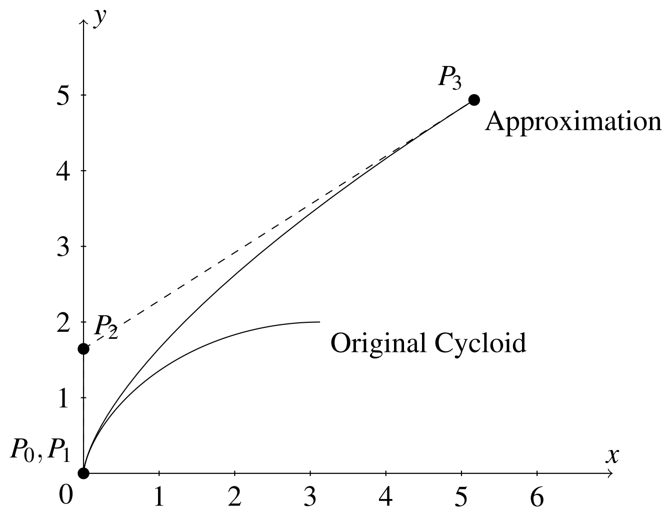

Figure 6.3: Approximation using Maclaurin series

This shws that the approximated Bézier curve will tangentially meet the original cycloid at $t = 0$, satisfying the boundary condition at the initial point. However, as the curves approach their terminal points, the error in approximation becomes very large. Also, the Bézier curve does not coincide with the curve nor have same slope of tangent at $t = 1$.

When error function of this approximation is plotted on a graph, it is clear that the maximum error occurs at $t = 1$, as shown in Fig. 6.7.

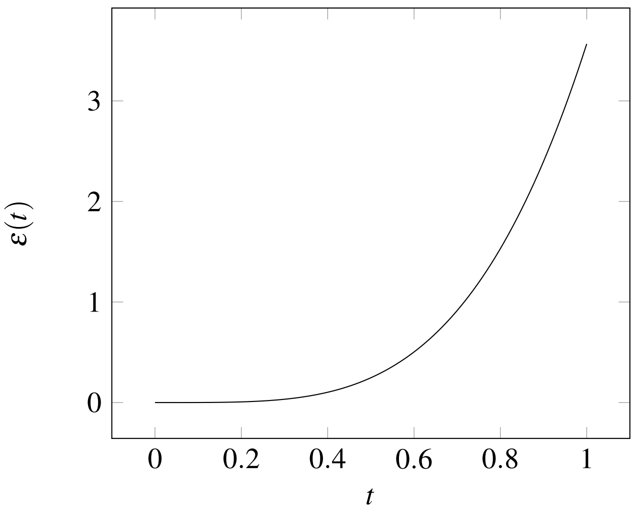

Figure 6.4: Error of the approximation using Maclaurin polynomial

A computation of error function at $t = 1$ yields that the maximum error is approximately 3.566.

6.2 Approximation of Trigonometric Function by Dokken et al. (1990)

Dokken et al gives an excellent Bézier approximation of a unit circle in their paper ”Good approximation of circles by curvature-continuous Bezier curves” from 1990. In fact, they were able to approximate a quarter sector with maximum error in radius of $1.4 * 10^{-4}$, which is a maximum of 0.014% error from the original circle. By taking the x and y components of their Bézier approximation, we can obtain a good approximation of cosine and sine functions, respectively.

Dokken et al. (1990) proved that a cubic Bézier curve with following control points has a maximum radial error of $\frac{1}{27}$ from a unit semicircle.

$$P_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + L \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$
$$P_2 = \begin{bmatrix} \cos\pi \\ \sin\pi \end{bmatrix} - L \begin{bmatrix} -\sin\pi \\ \cos\pi \end{bmatrix}, \qquad P_3 = \begin{bmatrix} \cos\pi \\ \sin\pi \end{bmatrix},$$

where $L = \frac{4}{3}\tan\frac{\pi}{4} = \frac{4}{3}$. Substituting the value of L and simplifying yields

$$P_0 = \begin{bmatrix}1\\0\end{bmatrix}, \qquad P_1 = \begin{bmatrix}1\\\frac{4}{3}\end{bmatrix},$$
$$P_2 = \begin{bmatrix}-1\\\frac{4}{3}\end{bmatrix}, \qquad P_3 = \begin{bmatrix}-1\\0\end{bmatrix},$$

As such, a cubic Bézier approximation of a unit semicircle is

$$\begin{aligned}x(t)(=\cos(\pi t)) &\approx (1-t)^3 + 3(1-t)^2t - 3(1-t)t^2 - t^3,\\ y(t)(=\sin(\pi t)) &\approx 4(1-t)^2t + 4(1-t)t^2.\end{aligned}$$

By substituting the x and y components of the above Bézier curve as approximations of trigonometric functions in the parametric equations of a cycloid, an approximation of a cycloid can be obtained as following.

$$\begin{aligned}x(t) &= \pi t - \sin(\pi t)\\ &\approx \pi t - (4(1-t)^2t + 4(1-t)t^2)\\ &= \pi t - 4(t - 2t^2 + t^3) - 4(t^2 - t^3)\\ &= \pi t - 4t + 8t^2 - 4t^3 - 4t^2 + 4^3\\ &= (\pi - 4)t + 4t^2.\end{aligned}$$

$$\begin{aligned}y(t) &= 1 - \cos(\pi t)\\ &\approx 1 - ((1-t)^3 + 3(1-t)^2t - 3(1-t)t^2 - t^3)\\ &= 1 - (1-t)^3 - 3(1-t)^2t + 3(1-t)t^2 + t^3\\ &= 1 - (1 - 3t + 3t^2 - t^3) - 3(t - 2t^2 + t^3) + 3(t^2 - t^3) + t^3\\ &= 1 - 1 + 3t - 3t^2 + t^3 - 3t + 6t^2 - 3t^3 + 3t^2 - 3t^3 + t^3\\ &= 6t^2 - 4t^3.\end{aligned}$$

In order to put this parametric equations into a Bézier form, we will follow the similar process of simplifying the equation of a Bézier curve with respect to t and equating the coefficients.

Since

$$\begin{aligned}b(t) &= \sum_{i=0}^{3} P_i \binom{3}{i} (1-t)^{3-i}t^i.\\ &= P_0(1 - 3t + 3t^2 + t^3) + 3P_1(t - 2t^2 + t^3) + 3P_2(t^2 - t^3) + P_3t^3\\ &= P_0 + 3(P_1 - P_0)t + 3(P_2 - 2P_1 + P_0)t^2 + (P_3 - 3P_2 + 3P_1 - P_0)t^3,\end{aligned}$$

the following 4 simultaneous equations for the x component of the control points can be produced by equating the coefficients with that of the approximation above.

$$\begin{aligned}
P_{0,x} &= 0, \\
3(P_{1,x} - P_{0,x}) &= \pi - 4, \\
3(P_{2,x} - 2P_{1,x} + P_{0,x}) &= 4, \\
P_{3,x} - 3P_{2,x} + 3P_{1,x} - P_{0,x} &= 0,
\end{aligned}$$

Solving the simultaneous equations yields

$$\begin{aligned}
\therefore P_{0,x} &= 0, \\
\therefore P_{1,x} &= \frac{\pi - 4}{3}, \\
\therefore P_{2,x} &= \frac{2\pi - 4}{3}, \\
\therefore P_{3,x} &= \pi,
\end{aligned}$$

Similarly, another set of 4 simultaneous equations can be produced for the y component of the control points.

$$\begin{aligned}
P_{0,x} &= 0, \\
3(P_{1,x} - P_{0,x}) &= 0, \\
3(P_{2,x} - 2P_{1,x} + P_{0,x}) &= 6, \\
P_{3,x} - 3P_{2,x} + 3P_{1,x} - P_{0,x} &= -4,
\end{aligned}$$

Solving the simultaneous equations yields

$$\begin{aligned}
\therefore P_{0,x} &= 0, \\
\therefore P_{1,x} &= 0, \\
\therefore P_{2,x} &= 2, \\
\therefore P_{3,x} &= 2,
\end{aligned}$$

As such, the control points of the cubic Bézier approximation of cycloid using Dokken et al.'s approximation of unit semicircle are

$$\begin{aligned}
P_0 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & P_1 &= \begin{bmatrix} \frac{\pi-4}{3} \\ 0 \end{bmatrix}, \\
P_2 &= \begin{bmatrix} \frac{2\pi-4}{3} \\ 2 \end{bmatrix}, & P_3 &= \begin{bmatrix} \pi \\ 2 \end{bmatrix},
\end{aligned}$$

Graphing the approximation and its error yields the following:

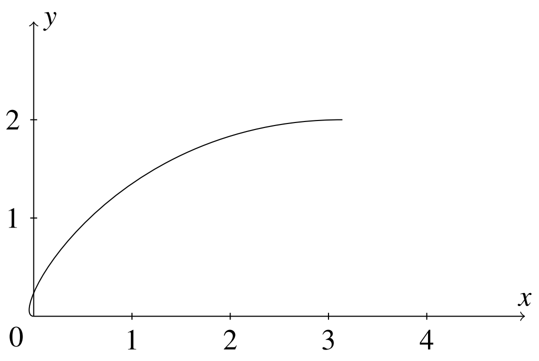

Figure 6.5: Bézier approximation using unit semicircle approximation by Dokken et al. (1990)

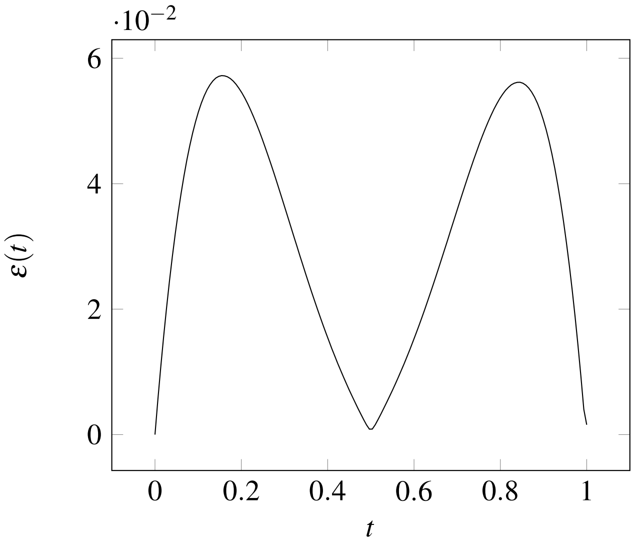

Figure 6.6: Error of the approximation

Fig 6.9 shows that the maximum error of the approximation is 0.0575, which occurs twice at $t = 0.157$ and $t = 0.843$. Compared to the previous approximation, this is clearly much closer to the original curve. Moreover, it coincides with the original curve at $t = 0$ and $t = 1$, and the slope of the tengent line at $t = 1$ is 0 (horizontal), which is equal to that of the original curve. However, at $t = 0$, the tangent lines of the two curves are perpendicular to each other. The cycloid has a vertical tangent, whereas, because the control points P_0, P_1 form a flat horizontal line, the Bézier approximation has a horizontal curve. This is shown more clearly in Fig. 6.10.

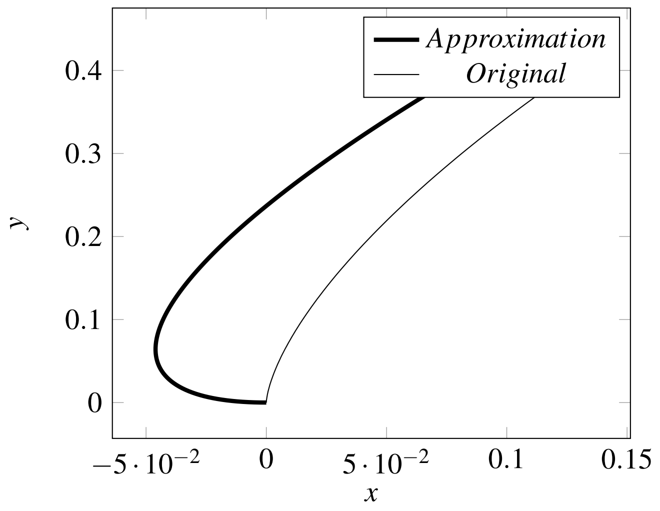

Figure 6.7: Error of the approximation

As such, this approximation fails to tangentially coincide the original cycloid at $t = 0$, slightly harming its accuracy.

6.3 Iterative Numerical Approximation

With the help of numerical analysis using a computer, a very close approximation can be made. However, in order to do so, we must exploit the properties of Bézier curves.

We know that the initial and terminal control points (P_0 and P_3, respectively) coincides with the interpolated point of the Bézier curve will be at $t = 0$ and $t = 1$, respectively. Moreover, the second and third control points (P_1 and P_2, respectively) will lay on the tangent of the original curve at $t = 0$ and $t = 1$ if the approximation were to tangentially coincide with the original curve at the initial and terminal points. Graphically, this can be represented as shown in Fig. 6.11.

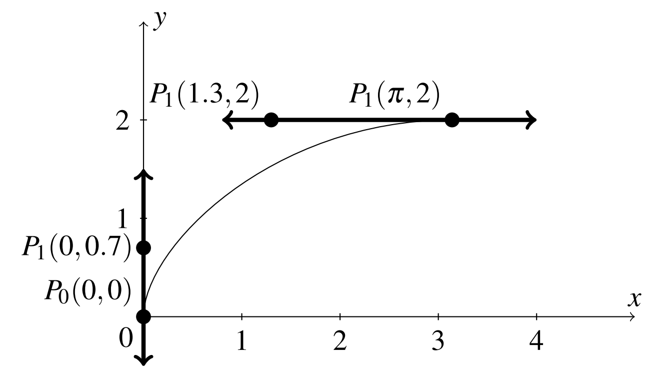

Figure 6.8: Possible arrangement of the control points

As such, the control points will be as following:

$$P_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ u \end{bmatrix},$$
$$P_2 = \begin{bmatrix} \pi \\ 2 \end{bmatrix} - \begin{bmatrix} v \\ 0 \end{bmatrix}, \qquad P_3 = \begin{bmatrix} \pi \\ 2 \end{bmatrix},$$

where $u, v \in \mathbb{R}^+$. As such, finding values of u and v that will minimize the maximum error $\max_{t \in [0,1]} \varepsilon(t)$ will give a close approximation of a cycloid with a cubic Bézier curve that fits the criteria of initial and terminal points discussed before.

Let $\zeta(u, v)$ be a function that gives that maximum error of Bézier approximation for given values u and v, or

$$\zeta(u, v) = \max_{t \in [0,1]} \varepsilon(t). \tag{15}$$

As such, $\arg\min_{u,v}(\zeta(u, v))$ will yield values u, v that will give the best possible approximation. Graphing $\zeta(u, v)$ using GNU Octave yields the following:

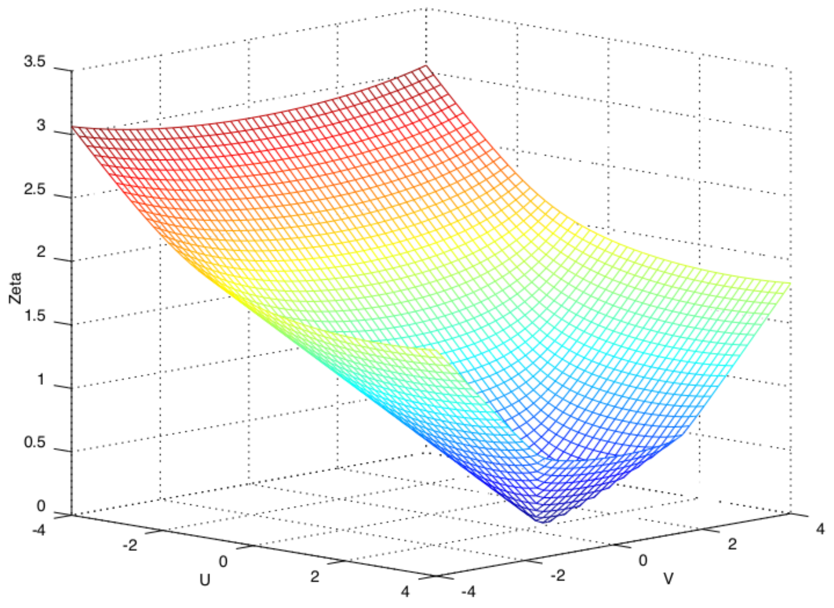

Figure 6.9: Plot of $u, v \in [-4, 4]$ and $\zeta(u, v)$

Using GNU Octave, $\underset{u,v}{\arg\min}(\zeta(u, v))$ is approximated iteratively, with initial domains of $u \in [0, 2]$ and $v \in [1, 3]$, in which each iteration subdivides the initial interval into 10, finds the minimum, and finds the minimum around the u, v discovered from previous iteration with a domain of $\pm$ the previous interval. The 10 subdivision is chosen because it is enough to ensure that the iteration will converge into the minimum within the initial domains, which can be deduced by examining Fig. 6.12. For the specific programming code, see Appendix A.

After 10 iteration, the approximation converged to $u = 0.000988889$ and $v = 2.50477$ with $\zeta(u, v) = 0.0809535$. As such, the control points of the Bézier approximation are

$$P_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 0 \\ 0.000988889 \end{bmatrix},$$
$$P_2 = \begin{bmatrix} \pi - 2.50477 \\ 2 \end{bmatrix}, \qquad P_3 = \begin{bmatrix} \pi \\ 2 \end{bmatrix}.$$

Graphing the Bézier approximation yields the following:

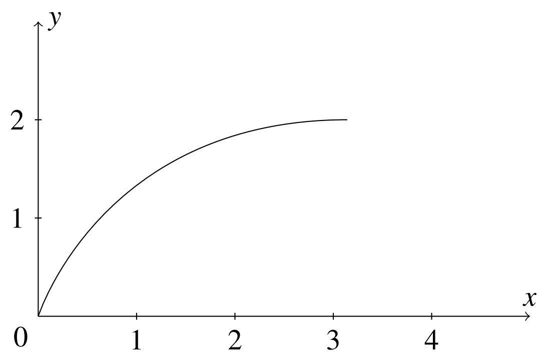

Figure 6.10: Bézier approximation using unit semicircle approximation by Dokken et al. (1990)

Although this appears to closely approximate the original cycloid, it is not as good as the previous approximation with Bézier semicircle of Dokken et al. (1990), which had $\max_{t\in[0,1]} \varepsilon(t)$ of 0.0575, compared to $\approx$ 0.0810 of this approximation. However, this approximation fits the boundary conditions of tangentially coinciding with the original cycloid at $t = 0$ and $t = 1$.

7 Evaluation

Table 7.1 shows a comparison between the three methods of approximation.

Method	Meet cycloid at $t = 0$	Same tangent as cycloid at $t = 0$	Meet cycloid at $t = 1$	Same tangent as cycloid at $t = 0$	$\max_{t \in [0,1]} \varepsilon(t)$
Maclaurin Polynomial	Yes	Yes	No	No	3.566
Semicircle Approximation by Dokken et al. (1990)	Yes	No	Yes	Yes	0.0575
Numerical Analysis	Yes	Yes	Yes	Yes	0.0810

Table 7.1: Comparison of Bézier approximations

Despite the approximation from numerical analysis having met all the boundary conditions, it still fails to give the least error among the three. This is interesting, because it is shown that the satisfying the boundary conditions will not necessarily provide the best approximation. As such, the results show that, if the Bézier curve was to be used to approximate cycloids, a significant error is inevitable no matter which method of approximation were to be used. Moreover, the traditional method of using Maclaurin polynomial for approximation of a non-polynomial curve is shown to be not appropriate for Bézier approximations, unless much a Bézier curve of much higher degree is used. Overall, Bézier curve appears to be a good approximant of a cycloid to some extent.

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*.

Dokken, T., Daehlen, M., Lyche, T., & Morken, K. (1990). Good approximation of circles by curvature-continuous Bézier curves. *Computer Aided Geometric Design*, *7*(1), 33 - 41. doi: https://doi.org/10.1016/0167-8396(90)90019-N

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.

A GNU Octave Code for Numerical Analysis

```
function extended_essay()

   [u, v, minZeta] = approximate(0, 2, 1, 3, 0.1);

   printf("u: %d\n", u);

   printf("v: %d\n", v);

   printf("min(zeta(u,v)): %d\n", minZeta);

   printf("\n");


   if (u <= 0)

     u = 0.1;

   endif


   for i = 1:9

     [u, v, minZeta] = approximate(u-10^(-i), u+10^(-i), v-10^(-i), v+10^(-i), 10^(-i-1));

     printf("u: %d\n", u);

     printf("v: %d\n", v);

     printf("min(zeta(u,v)): %d\n", minZeta);

     printf("\n");


     if (u <= 0)

       u = 10^(-i-1);

     endif

   endfor

endfunction


function retval = cycloid_x(t)

  retval = pi * t - sin(pi * t);

endfunction
```

```
function retval = cycloid_y(t)
  retval = 1 - cos(pi * t);
endfunction

function retval = bezier(t, P0, P1, P2, P3)
  retval = P0 .* (1-t).^3 + 3 .* P1 .* (1-t).^2 .* t + 3 .* P2 .* (1-t) .* t.^2 + P3 .* t.^3;
endfunction

function retval = epsilon(t, P0, P1, P2, P3) % t is scalar value, and P0~P3 are 2D vector.
  Bx = bezier(t, P0(1), P1(1), P2(1), P3(1));
  By = bezier(t, P0(2), P1(2), P2(2), P3(2));

  retval = sqrt((cycloid_x(t)-Bx).^2 + (cycloid_y(t)-By).^2);
endfunction

function retval = zeta(u, v) % Using t interval of 0.001
  t = 0:0.001:1;
  retval = max(epsilon(t, [0, 0], [0, u], [pi - v, 2], [pi, 2]));
endfunction

function [u, v, minZeta] = approximate(u_start, u_end, v_start, v_end, interval)
  minZeta = inf;
  u = 0;
  v = 0;

  for i = u_start:interval:u_end
    for j = v_start:interval:v_end
      curZeta = zeta(i, j);
      if (curZeta < minZeta)
        minZeta = curZeta;
        u = i;
        v = j;
      endif
    endfor
  endfor
endfunction
```