Different Approaches of Approximating a Cycloid Using Bézier Curves

To what extent can the Bézier curve be used to approximate a cycloid?

Mathematics

Word count: 3345

Contents

1. Introduction

In the modern era, computer has been an influential tool of humans. In fact, generations born after the 2000s, including myself, have hardly lived an era with absence of heavy influence of computers. Many of their powerful abilities, such as being able to render high quality realistic images, are taken for granted by many people. A computer 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, the algorithms used for the computers to portray smooth, curved objects in such a swift manner is hard to imagine at a first glance.

This swift rendering of curves and curved surfaces is achieved using many methods, but one of the most prominent method is the Bézier curve. Bézier curves are 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). However, because the Bézier curve is a parametric polynomial curve, it cannot give an exact, mathematically equivalent representation of non-polynomial curves, such as circles, ellipses, and graphs of sine and cosine functions. As such, the topic of approximating various kinds of non-polynomial curves using the Bézier curve has been thoroughly investigated along with the development of computer graphics. One example of this is the unit circle approximation given by Dokken, Daehlen, Lyche, and Morken (1990), which approximates a unit circle with a maximum error in radius of $1.4 * 10^{-4}$ with only 4 piecewise Bézier curves. Yet, after a thorough research, it was evident that there is no formal attempt of approximation done for cycloids using a Bézier curve. Cycloids are parametric curves that are formed by tracing one point on a circle that is rolling on a flat surface. This was quite an unexpected finding, because cycloids themselves are defined parametrically like Bézier curves, which makes the approximation quite natural. As such, I have determined to **investigate on how, and to what extent, a Bézier curve can be used to approximate a cycloid**.

2. Recursive Linear Interpolation

Straight lines are one of the easiest shapes one can imagine and draw, and such holds true even for computers. In the context of computer graphics, drawing straight lines is one of the most rudimentary job a computer can perform. Despite their simplicity, a thorough knowledge about the process used to describe and draw straight lines is required to understand the inner-workings of the Bézier curve.

Such rendering of straight lines are achieved by using linear interpolation. Linear interpolation is formally defined as a computation of a point on a straight path between two points $\boldsymbol{P}_A$ and $\boldsymbol{P}_B$ given a value $t \in [0,1]$ (Prautzsch, Boehm, & Paluszny, 2013). Let us imagine that a driver is on a journey through a straight road. If $\boldsymbol{P}_A$ and $\boldsymbol{P}_B$ were to be the starting point and the destination of a driver, the ratio between the distance in which a driver has driven, and the distance he/she will be driving would be $t : (1-t)$. This straight path between $\boldsymbol{P}_A$ and $\boldsymbol{P}_B$ is called the interpolant. This is shown in Fig. 2.1. Hereinafter, all of the figures in this paper are produced by myself using Latex's PGF-Tikz unless specified otherwise.

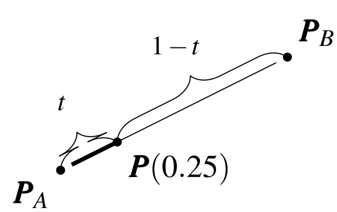

Figure 2.1: Comparison of t and $1-t$ for $t = 0.25$

Mathematically, the position of the driver can be expressed as the following vector equation:

$$\boldsymbol{P}(t) = \boldsymbol{P}_A + (\boldsymbol{P}_B - \boldsymbol{P}_A)t. \tag{1}$$

Rearranging the equation as following emphasizes the significance of t and $(1-t)$.

$$\boldsymbol{P}(t) = \boldsymbol{P}_A(1-t) + \boldsymbol{P}_B t. \tag{2}$$

This linear interpolation can be performed with 3 points. If we have three points $\boldsymbol{P}_0 = (1,1)$, $\boldsymbol{P}_1 = (2,2)$, and $\boldsymbol{P}_2 = (3,1)$, we can produce straight lines between these three points using the equation of linear interpolation $\boldsymbol{P}(t) = \boldsymbol{P}_A(1-t) + \boldsymbol{P}_B t$ as Fig. 2.2 shown below.

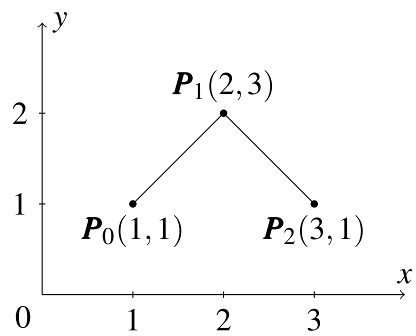

Figure 2.2: Linear interpolants between $\boldsymbol{P}_0, \boldsymbol{P}_1$, and $\boldsymbol{P}_1, \boldsymbol{P}_2$

The interpolated point between $\boldsymbol{P}_0$ and $\boldsymbol{P}_1$ is notated $\boldsymbol{P}_{0...1}$, which is $(1.5, 1.5)$ at $t = 0.5$. Similarly, $\boldsymbol{P}_{1...2}$ is $(2.5, 1.5)$ at $t = 0.5$. These newly interpolated points can be used as the starting and the terminal points of new linear interpolation, as shown in Fig. 2.3.

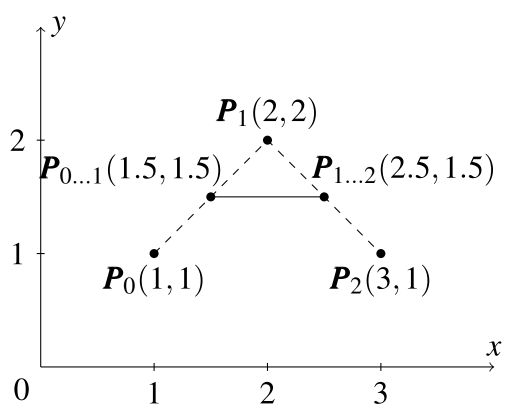

Figure 2.3: Linear interpolant between $\boldsymbol{P}_{0...1}, \boldsymbol{P}_{1...2}$

Finally, let $\boldsymbol{P}_{0...2}$ be the linearly interpolated point between $\boldsymbol{P}_{0...1}$ and $\boldsymbol{P}_{1...2}$ at $t = 0.5$. Then, $\boldsymbol{P}_{0...2}$ is a resulting point of the quadratic interpolation from points $\boldsymbol{P}_0$, $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ at $t = 0.5$, as shown in Fig. 2.4.

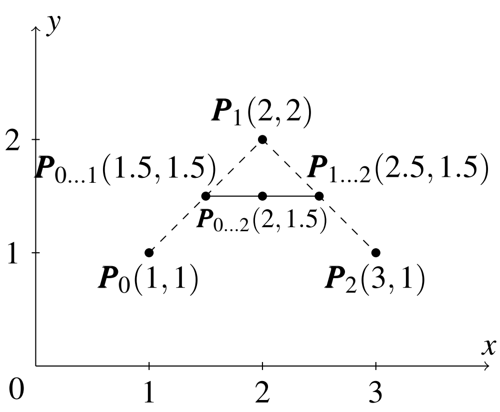

Figure 2.4: Quadratic interpolation at $t = 0.5$

If we apply the above linear interpolation repeatedly with $t = 0.25$ and $t = 0.75$, then we will have points shown in Fig. 2.5 and 2.6.

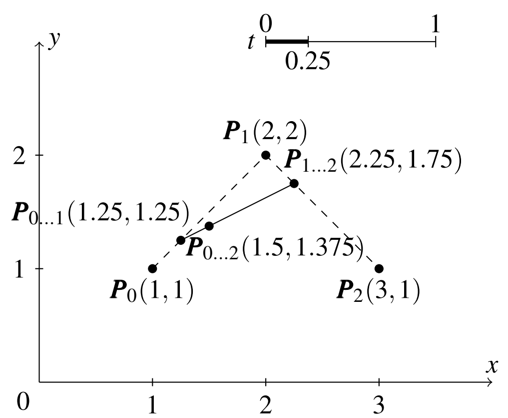

Figure 2.5: Quadratic interpolation at $t = 0.25$

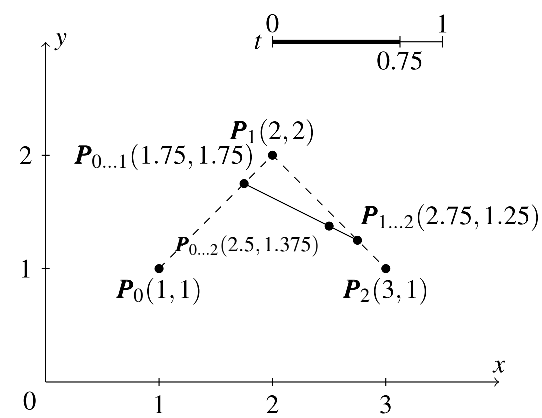

Figure 2.6: Quadratic interpolation at $t = 0.75$

Using a computer with small enough interval of t, the quadratic interpolant can be constructed and rendered smoothly, as shown in Fig. 2.7.

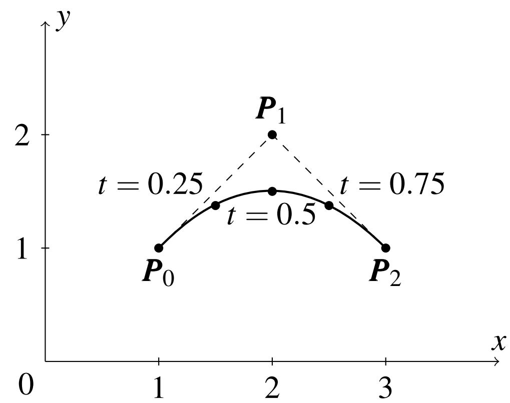

Figure 2.7: Quadratic interpolant

This process of repeated linear interpolation is called 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”, meaning that it repeats itself to meet the objective (Prautzsch et al., 2013). 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 $\boldsymbol{P}_0, \boldsymbol{P}_1$ and $\boldsymbol{P}_2$ of the previous example. As such, they are the ones that control 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 Bézier curves with three control points. However, by following the similar process of repeating linear interpolation, curves with higher number of control points can be constructed. For example, let the control points be $\boldsymbol{P}_0 = (1,1), \boldsymbol{P}_1 = (1,5), \boldsymbol{P}_2 = (5,5)$ and $\boldsymbol{P}_3 = (5,1)$. A set of linear interpolants is produced as shown in Fig. 2.8.

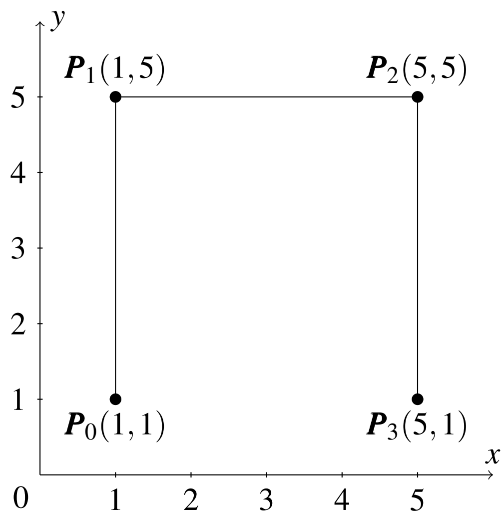

Figure 2.8: Linear interpolants between the four Bézier control points

Firstly, the points $\boldsymbol{P}_{0\ldots1}, \boldsymbol{P}_{1\ldots2}$ and $\boldsymbol{P}_{2\ldots3}$ are interpolated. These points are used to produce another set of linear interpolants, as shown in Fig. 2.9.

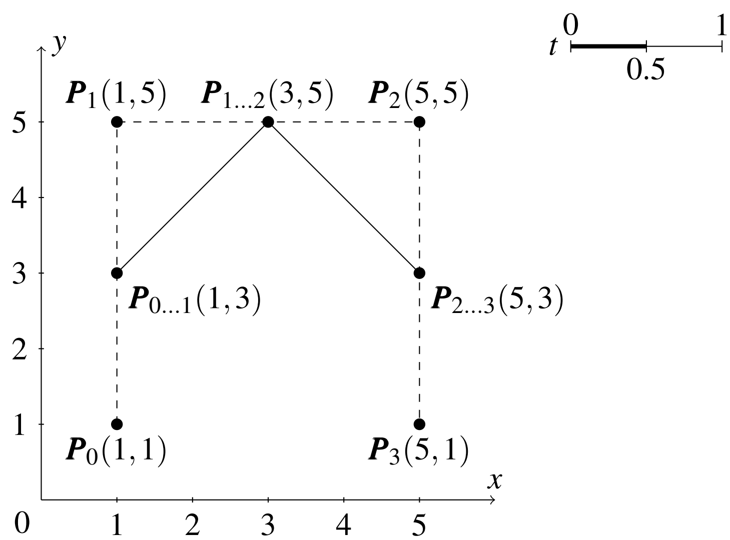

Figure 2.9: Second recursion of De Casteljau algorithm at $t = 0.5$

Then, the points $\boldsymbol{P}_{0\ldots1}, \boldsymbol{P}_{1\ldots2}$ and $\boldsymbol{P}_{2\ldots3}$ are used to produce points $\boldsymbol{P}_{0\ldots2}$ and $\boldsymbol{P}_{1\ldots3}$, as shown in Fig. 2.10.

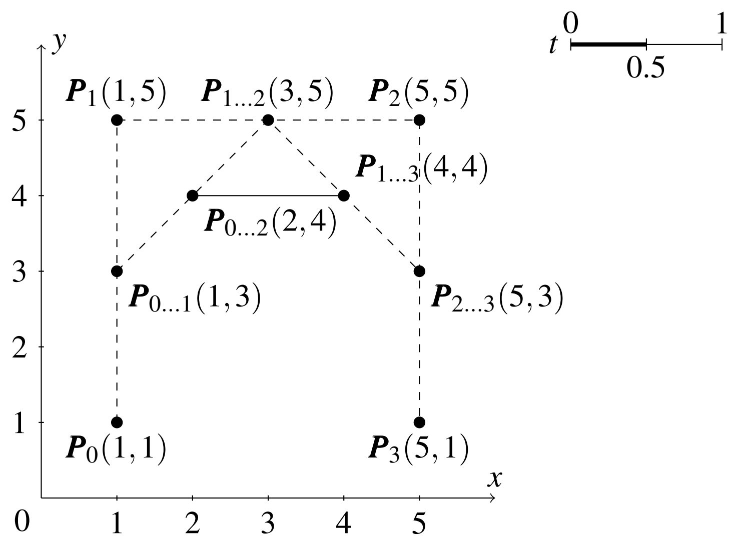

Figure 2.10: Third recursion of De Casteljau algorithm at $t = 0.5$

Finally, the resulting point, $\boldsymbol{P}_{0\ldots3}$, of the cubic interpolation at $t = 0.5$ is obtained from points $\boldsymbol{P}_{0\ldots2}$ and $\boldsymbol{P}_{1\ldots3}$, as shown in Fig. 2.11:

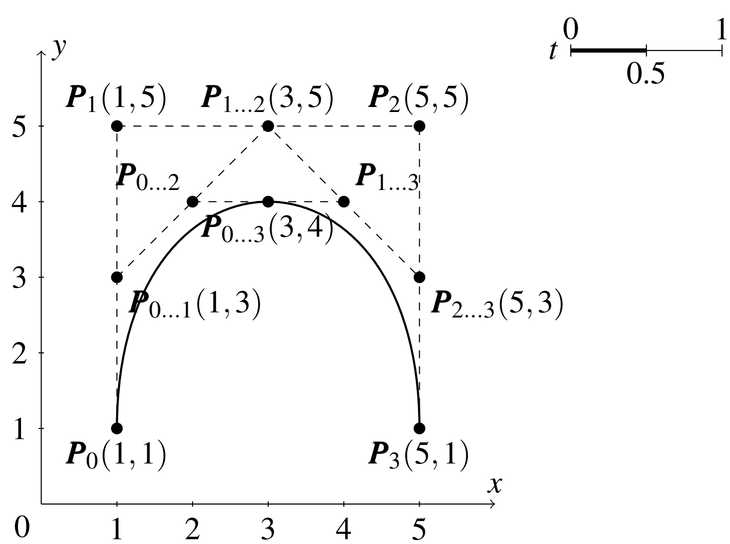

Figure 2.11: Resulting point of the cubic interpolation at $t = 0.5$ and the cubic interpolant

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

1. Interpolate a set of points corresponding to a value $t \in [0, 1]$ 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 only been vaguely mentioned, and its mathematical definition is yet unclear. This algorithm can be used to deduce the formal definition of a Bézier curve (Prautzsch et al., 2013). Let $\boldsymbol{P}_{a...b}(t)$ be a function that gives an interpolated point from a set of control points $\boldsymbol{P}_a, \boldsymbol{P}_{a+1}, \boldsymbol{P}_{a+2}, \ldots, \boldsymbol{P}_{b-1}, \boldsymbol{P}_b$, where $a, b \in \mathbb{Z}_{\leq 0}, a \geq b$. If $a = b$, then $\boldsymbol{P}_{a...b}(t) = \boldsymbol{P}_a = \boldsymbol{P}_b$. Next, let the control points of an nth degree Bézier curve be $\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_{n-1}, \boldsymbol{P}_n$. Performing a set of linear interpolation to consecutive pairs of control points at t gives the following:

$$\begin{gathered}
\boldsymbol{P}_{0...1}(t) = \boldsymbol{P}_0(1-t) + \boldsymbol{P}_1 t, \\
\boldsymbol{P}_{1...2}(t) = \boldsymbol{P}_1(1-t) + \boldsymbol{P}_2 t, \\
\boldsymbol{P}_{2...3}(t) = \boldsymbol{P}_2(1-t) + \boldsymbol{P}_3 t, \\
\vdots \\
\boldsymbol{P}_{n-1...n}(t) = \boldsymbol{P}_{n-1}(1-t) + \boldsymbol{P}_n t.
\end{gathered}$$

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

$$\begin{gathered}
\boldsymbol{P}_{0...2}(t) = [\boldsymbol{P}_{0...1}(t)](1-t) + [\boldsymbol{P}_{1...2}(t)]t, \\
\boldsymbol{P}_{1...3}(t) = [\boldsymbol{P}_{1...2}(t)](1-t) + [\boldsymbol{P}_{2...3}(t)]t, \\
\boldsymbol{P}_{2...4}(t) = [\boldsymbol{P}_{2...3}(t)](1-t) + [\boldsymbol{P}_{3...4}(t)]t, \\
\vdots \\
\boldsymbol{P}_{n-2...n}(t) = [\boldsymbol{P}_{n-2...n-1}(t)](1-t) + [\boldsymbol{P}_{n-1...n}(t)]t.
\end{gathered}$$

Substituting the interpolation functions with their definitions,

$$\begin{gathered}
\boldsymbol{P}_{0...2}(t) = [\boldsymbol{P}_0(1-t) + \boldsymbol{P}_1 t](1-t) + [\boldsymbol{P}_1(1-t) + \boldsymbol{P}_2 t]t, \\
\boldsymbol{P}_{1...3}(t) = [\boldsymbol{P}_1(1-t) + \boldsymbol{P}_2 t](1-t) + [\boldsymbol{P}_2(1-t) + \boldsymbol{P}_3 t]t, \\
\boldsymbol{P}_{2...4}(t) = [\boldsymbol{P}_2(1-t) + \boldsymbol{P}_3 t](1-t) + [\boldsymbol{P}_3(1-t) + \boldsymbol{P}_4 t]t, \\
\vdots \\
\boldsymbol{P}_{n-2...n}(t) = [\boldsymbol{P}_{n-2}(1-t) + \boldsymbol{P}_{n-1} t](1-t) + [\boldsymbol{P}_{n-1}(1-t) + \boldsymbol{P}_n t]t.
\end{gathered}$$

Simplifying the equations above,

$$
\begin{aligned}
\boldsymbol{P}_{0\ldots2}(t) &= \boldsymbol{P}_0(1-t)^2+2\boldsymbol{P}_1(1-t)t+\boldsymbol{P}_2t^2,\\
\boldsymbol{P}_{1\ldots3}(t) &= \boldsymbol{P}_1(1-t)^2+2\boldsymbol{P}_2(1-t)t+\boldsymbol{P}_3t^2,\\
\boldsymbol{P}_{2\ldots4}(t) &= \boldsymbol{P}_2(1-t)^2+2\boldsymbol{P}_3(1-t)t+\boldsymbol{P}_4t^2,\\
&\vdots\\
\boldsymbol{P}_{n-2\ldots n}(t) &= \boldsymbol{P}_{n-2}(1-t)^2+2\boldsymbol{P}_{n-1}(1-t)t+\boldsymbol{P}_nt^2.
\end{aligned}
$$

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

$$
\begin{aligned}
\boldsymbol{P}_{0\ldots3}(t) &= \boldsymbol{P}_0(1-t)^3+3\boldsymbol{P}_1(1-t)^2t+3\boldsymbol{P}_2(1-t)t^2+\boldsymbol{P}_3t^3,\\
\boldsymbol{P}_{1\ldots4}(t) &= \boldsymbol{P}_1(1-t)^3+3\boldsymbol{P}_2(1-t)^2t+3\boldsymbol{P}_3(1-t)t^2+\boldsymbol{P}_4t^3,\\
\boldsymbol{P}_{2\ldots5}(t) &= \boldsymbol{P}_2(1-t)^3+3\boldsymbol{P}_3(1-t)^2t+3\boldsymbol{P}_4(1-t)t^2+\boldsymbol{P}_5t^3,\\
&\vdots\\
\boldsymbol{P}_{n-3\ldots n}(t) &= \boldsymbol{P}_{n-3}(1-t)^3+3\boldsymbol{P}_{n-2}(1-t)^2t+3\boldsymbol{P}_{n-1}(1-t)t^2+\boldsymbol{P}_nt^3.
\end{aligned}
$$

Repeating the same procedure again,

$$
\begin{aligned}
\boldsymbol{P}_{0\ldots4}(t) &= \boldsymbol{P}_0(1-t)^4+4\boldsymbol{P}_1(1-t)^3t+6\boldsymbol{P}_2(1-t)^2t^2+4\boldsymbol{P}_3(1-t)t^3+\boldsymbol{P}_4t^4,\\
\boldsymbol{P}_{1\ldots5}(t) &= \boldsymbol{P}_1(1-t)^4+4\boldsymbol{P}_2(1-t)^3t+6\boldsymbol{P}_3(1-t)^2t^2+4\boldsymbol{P}_4(1-t)t^3+\boldsymbol{P}_5t^4,\\
\boldsymbol{P}_{2\ldots6}(t) &= \boldsymbol{P}_2(1-t)^4+4\boldsymbol{P}_3(1-t)^3t+6\boldsymbol{P}_4(1-t)^2t^2+4\boldsymbol{P}_5(1-t)t^3+\boldsymbol{P}_6t^4,\\
&\vdots\\
\boldsymbol{P}_{n-4\ldots n}(t) &= \boldsymbol{P}_{n-4}(1-t)^4+4\boldsymbol{P}_{n-3}(1-t)^3t+6\boldsymbol{P}_{n-2}(1-t)^2t^2+4\boldsymbol{P}_{n-1}(1-t)t^3+\boldsymbol{P}_nt^4.
\end{aligned}
$$

Here, a pattern that imitates binomial expansion of $((1-t)+t)^n$ emerges, along with the control points multiplied as coefficients of each terms as noted by Prautzsch et al. (2013). Continuing on a similar manner, the complete equation of interpolated Bézier curve is obtained.

$$
\begin{aligned}
\boldsymbol{P}_{0\ldots n}(t) = \boldsymbol{P}_0\binom{n}{0}(1-t)^n+\boldsymbol{P}_1\binom{n}{1}(1-t)^{n-1}t+\boldsymbol{P}_2\binom{n}{2}(1-t)^{n-2}t^2\\
+\cdots+\boldsymbol{P}_k\binom{n}{k}(1-t)^{n-k}t^k+\cdots+\boldsymbol{P}_{n-1}\binom{n}{n-1}(1-t)t^{n-1}+\boldsymbol{P}_n\binom{n}{n}t^n,
\end{aligned}
\tag{3}
$$

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

$$
\boldsymbol{P}_{0\ldots n}(t)=\sum_{i=0}^{n}\boldsymbol{P}_i\binom{n}{i}(1-t)^{n-i}t^i \quad \text{(Prautzsch et al., 2013).} \tag{4}
$$

This, in fact, is the definition of the Bézier curve, which is denoted $b_n(t)$. As such, the mathematical development of Bézier curves from linear interpolation has been investigated so far. The following part of the essay will examine the properties of Bézier curves and why they are useful when approximating a cycloid.

3. Definition and Important Properties of Bézier Curves

Thus far, we have investigated on the mathematical development of Bézier curves from the De Casteljau algorithm. However, before we can deduce different approaches of approximating a cycloid, the knowledge of the definition and some important properties of Bézier curves must precede.

To reiterate, the definition of Bézier curves is

$$\boldsymbol{b_n}(t) = \boldsymbol{P}_{0\ldots n} = \sum_{i=0}^{n} \boldsymbol{P}_i \binom{n}{i} (1-t)^{n-1} t^i \quad \text{(Prautzsch et al., 2013).} \tag{5}$$

Although there are numerous properties of the Bézier curve, this investigation will mostly utilize the following two. The first property is that, at $t = 0$ and $t = 1$, the interpolated points coincide with the first and last Bézier control points, respectively. The second property is that the tangents at $t = 0$ and $t = 1$ pass through points $\boldsymbol{P}_0$, $\boldsymbol{P}_1$ and $\boldsymbol{P}_{n-1}$, $\boldsymbol{P}_n$, respectively (Prautzsch et al., 2013). In Fig. 3.1, the control points $\boldsymbol{P}_0$ and $\boldsymbol{P}_3$ explain the first property, and $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ lies on the tangents at $t = 0$ and $t = 1$ of the Bézier curve, respectively, which explains the second property.

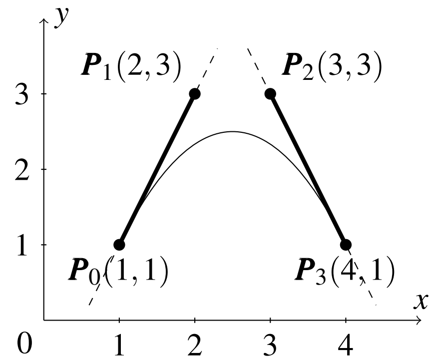

Figure 3.1: Tangent lines of a cubic Bézier curve

As such, two boundary conditions about the control points can be made for every approximation

1. The control points $\boldsymbol{P}_0$ and $\boldsymbol{P}_3$ coincides with the first and last points of the original curve, respectively; and
2. The control points $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ lie on the tangents at the first and last points of the original curve, respectively.

In other words, if the Bézier approximation was to meet the boundary conditions, it will **tangentially coincide** with the original curve at $t = 0$ and $t = 1$. These conditions will allow us to evaluate the quality of the approximation.

4. Bézier Approximation of a Cycloid

Bézier curve is a parametrically-defined curve, and one kind of a parametric curve is the cycloid. The following investigation develops a new method to approximate a cycloid using a Bézier curve, such as the iterative numerical analysis. This method is compared with the traditional methods, such as the Maclaurin polynomial, and the approximation of unit circle by Dokken et al. (1990).

A cycloid is a parametric curve that is defined by the following:

$$
\begin{aligned}
x(\theta) &= \theta - \sin\theta, \\
y(\theta) &= 1 - \cos\theta.
\end{aligned}
\tag{6}
$$

This is a curve that is constructed by plotting the path in which a point on a unit circle that is rolling on a smooth, straight surface passes through. Graphically, it is illustrated by Fig. 4.1.

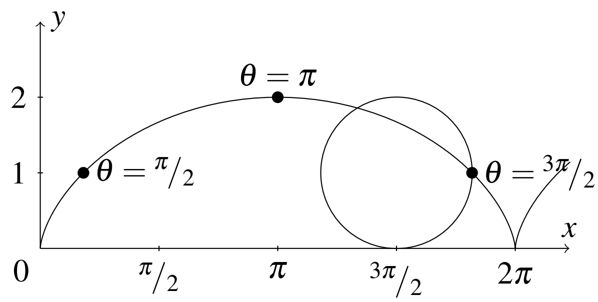

Figure 4.1: An arc of a cycloid

Because this curve infinitely repeats an identical arc, 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. 4.2.

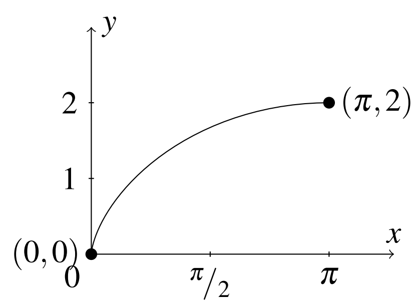

Figure 4.2: Section of cycloid for $\theta \in [0, \pi]$

Since a Bézier approximation is defined for $t \in [0, 1]$, the cycloid can be redefined as following with respect to variable t so that the arc shown in Fig 4.2 will be constructed for $t \in [0, 1]$.

$$
\begin{aligned}
x(t) &= \pi t - \sin \pi t, \\
y(t) &= 1 - \cos \pi t,
\end{aligned}
\tag{7}
$$

where their derivatives are

$$
\begin{aligned}
x(t) &= \pi - \pi \cos \pi t, \\
y(t) &= \pi \sin \pi t.
\end{aligned}
\tag{8}
$$

The three methods of approximation that are investigated are based on the Maclaurin polynomial, approximation of unit semicircle given by Dokken et al. (1990), and iterative numerical analysis. Although these are not the only methods of approximating a non-polynomial curve with a Bézier curve, they allow a thorough holistic judgement of Bézier curve as an approximant of a cycloid.

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

When evaluating an approximation, two approaches are used: its extent of tangential coincidence at initial and terminal points, and numerical error analysis.

Fig. 4.2 shows that the initial and terminal points of the cycloid are $(0,0)$ and $(\pi,2)$, respectively. Moreover, the computing 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 $t = 0$,

$$\frac{dy}{dx}|_{t=0} = \lim_{t\to 0^+} \frac{\pi \sin(\pi t)}{\pi - \pi \cos(\pi t)}$$
$$= \lim_{t\to 0^+} \frac{\pi^2 \cos(\pi t)}{\pi^2 \sin(\pi t)}$$

The limit above approaches positive infinity, which shows that the cycloid has a vertical tangent at $t = 0$. At the terminal point $t = 1$,

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

As such, assessing if the approximated curve tangentially coincides with the original cycloid at $t = 0$ and $t = 1$ allows us an assessment of the quality of the approximation to some extent.

For the error analysis, an error function $\varepsilon(t)$ is used, which yields a numerical value of the accuracy of an approximation, allowing a quantitative comparison between the three approximations. Using the Pythagoras theorem, the Euclidean distance between the cycloid and the Bézier approximation at $t \in [0,1]$ can be computed as following:

$$\varepsilon(t) = \sqrt{(x(t) - b_x(t))^2 + (y(t) - b_y(t))^2}, \tag{9}$$

where $x(t)$ and $y(t)$ are the x and y components of the parametric equations of a cycloid, and $b_x(t)$ and $b_y(t)$ are the x and y components of the Bézier approximation. Thus, by computing $\max\limits_{t\in[0,1]} \varepsilon(t)$, the maximum error of an approximation is evaluated, which gives us another way to assess the quality of an approximation to some degree.

4.1. Approximation Using the Maclaurin Polynomials

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

$$f(x) \approx \boldsymbol{P}_n(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!} + \ldots + \frac{f^{(n)}(0)}{n!} \tag{10}$$

Using the Maclaurin polynomial, the parametric equations of a cycloid can be approximated. Because a cubic Bézier curve is used, a Maclaurin polynomial with order of 3 is 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 a cycloid. To rearrange the Maclaurin polynomial into a Bézier form, we first expand and simplify the definition of Bézier curve into a polynomial in terms of t.

$$\begin{aligned}
\boldsymbol{b}(t) &= \sum_{i=0}^{3} \boldsymbol{P}_i \binom{3}{i} (1-t)^{3-i} t^i \\
&= \boldsymbol{P}_0(1-t)^3 + 3\boldsymbol{P}_1(1-t)^2 t + 3\boldsymbol{P}_2(1-t)t^2 + \boldsymbol{P}_3 t^3 \\
&= \boldsymbol{P}_0(1 - 3t + 3t^2 + t^3) + 3\boldsymbol{P}_1(t - 2t^2 + t^3) + 3\boldsymbol{P}_2(t^2 - t^3) + \boldsymbol{P}_3 t^3 \\
&= \boldsymbol{P}_0 - 3\boldsymbol{P}_0 t + 3\boldsymbol{P}_0 t^2 - \boldsymbol{P}_0 t^3 + 3\boldsymbol{P}_1 t - 6\boldsymbol{P}_1 t^2 + 3\boldsymbol{P}_1 t^3 + 3\boldsymbol{P}_2 t^2 - 3\boldsymbol{P}_2 t^3 + \boldsymbol{P}_3 t^3 \\
&= \boldsymbol{P}_0 + 3(\boldsymbol{P}_1 - \boldsymbol{P}_0)t + 3(\boldsymbol{P}_2 - 2\boldsymbol{P}_1 + \boldsymbol{P}_0)t^2 + (\boldsymbol{P}_3 - 3\boldsymbol{P}_2 + 3\boldsymbol{P}_1 - \boldsymbol{P}_0)t^3
\end{aligned}$$

Since

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

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

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

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

$$\boldsymbol{P}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \boldsymbol{P}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$\boldsymbol{P}_2 = \begin{bmatrix} 0 \\ \frac{\pi^2}{6} \end{bmatrix}, \qquad \boldsymbol{P}_3 = \begin{bmatrix} \frac{\pi^3}{6} \\ \frac{\pi^2}{2} \end{bmatrix}.$$

This approximation is shown in Fig. 4.3.

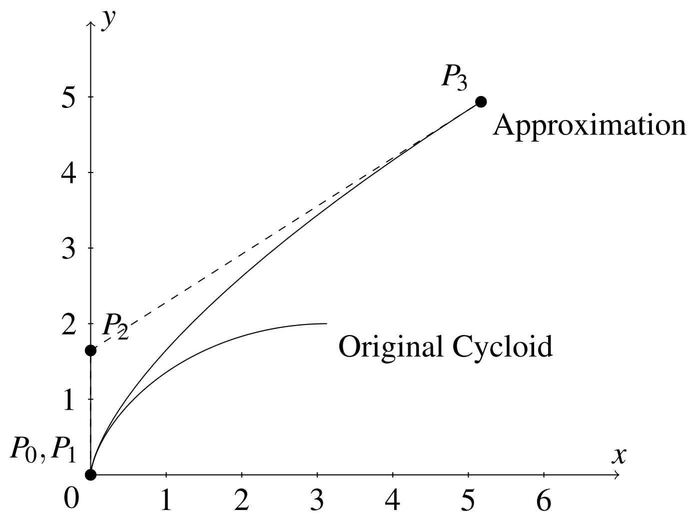

Figure 4.3: Bézier approximation using the Maclaurin polynomials

This shows 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 tangentially coincide with the original cycloid 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. 4.4.

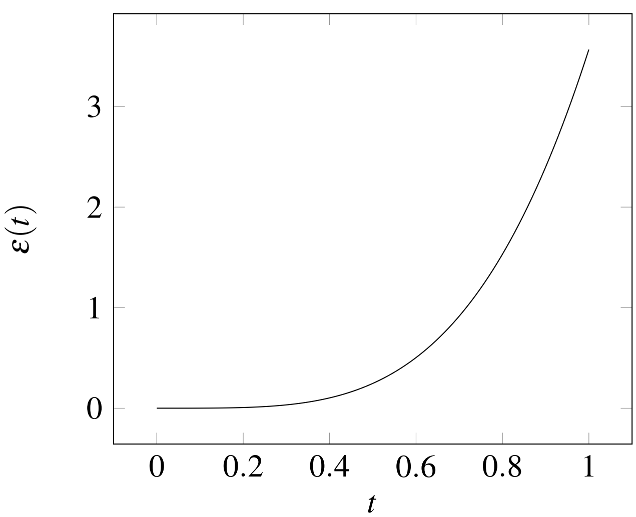

Figure 4.4: Error of the approximation using Maclaurin polynomial

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

4.2. Approximation Using Bézier Approximation of Unit Semicircle by Dokken et al. (1990)

Dokken et al. (1990) gives an excellent Bézier approximation of a unit circle in their paper ”Good approximation of circles by curvature-continuous Bezier curves” (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 unit circle. By taking the x and y components of their Bézier approximation, we can obtain good approximations 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 $\frac{1}{27}$, or approximately 0.0370 from a unit semicircle.

$$\begin{aligned} \boldsymbol{P}_0 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}, & \boldsymbol{P}_1 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + L \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \\ \boldsymbol{P}_2 &= \begin{bmatrix} \cos\pi \\ \sin\pi \end{bmatrix} - L \begin{bmatrix} -\sin\pi \\ \cos\pi \end{bmatrix}, & \boldsymbol{P}_3 &= \begin{bmatrix} \cos\pi \\ \sin\pi \end{bmatrix}, \end{aligned}$$

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

$$\begin{aligned} \boldsymbol{P}_0 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}, & \boldsymbol{P}_1 &= \begin{bmatrix} 1 \\ \frac{4}{3} \end{bmatrix}, \\ \boldsymbol{P}_2 &= \begin{bmatrix} -1 \\ \frac{4}{3} \end{bmatrix}, & \boldsymbol{P}_3 &= \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \end{aligned}$$

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

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

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

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

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

In order to put these 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 as done in the previous approximation.

Since

$$
\begin{aligned}
\boldsymbol{b}(t) &= \sum_{i=0}^{3} \boldsymbol{P}_i \binom{3}{i} (1-t)^{3-i} t^i. \\
&= \boldsymbol{P}_0(1 - 3t + 3t^2 + t^3) + 3\boldsymbol{P}_1(t - 2t^2 + t^3) + 3\boldsymbol{P}_2(t^2 - t^3) + \boldsymbol{P}_3 t^3 \\
&= \boldsymbol{P}_0 + 3(\boldsymbol{P}_1 - \boldsymbol{P}_0)t + 3(\boldsymbol{P}_2 - 2\boldsymbol{P}_1 + \boldsymbol{P}_0)t^2 + (\boldsymbol{P}_3 - 3\boldsymbol{P}_2 + 3\boldsymbol{P}_1 - \boldsymbol{P}_0)t^3,
\end{aligned}
$$

the following 4 simultaneous equations for the x component of the control points can be produced by equating the coefficients with the approximation of $x(t)$.

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

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

$$
\boldsymbol{P}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \boldsymbol{P}_1 = \begin{bmatrix} \frac{\pi-4}{3} \\ 0 \end{bmatrix},
$$
$$
\boldsymbol{P}_2 = \begin{bmatrix} \frac{2\pi-4}{3} \\ 2 \end{bmatrix}, \qquad \boldsymbol{P}_3 = \begin{bmatrix} \pi \\ 2 \end{bmatrix},
$$

Graphing the approximation and its error yields the following:

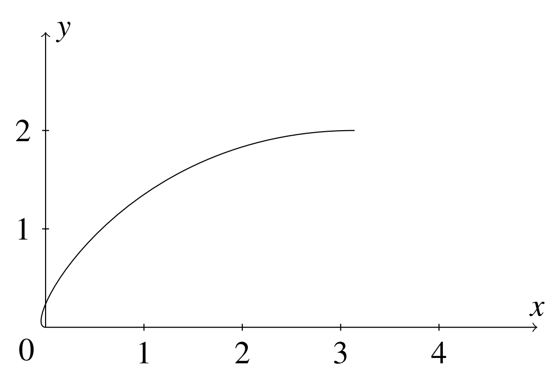

Figure 4.5: Bézier approximation using unit semicircle approximation by Dokken et al. (1990)

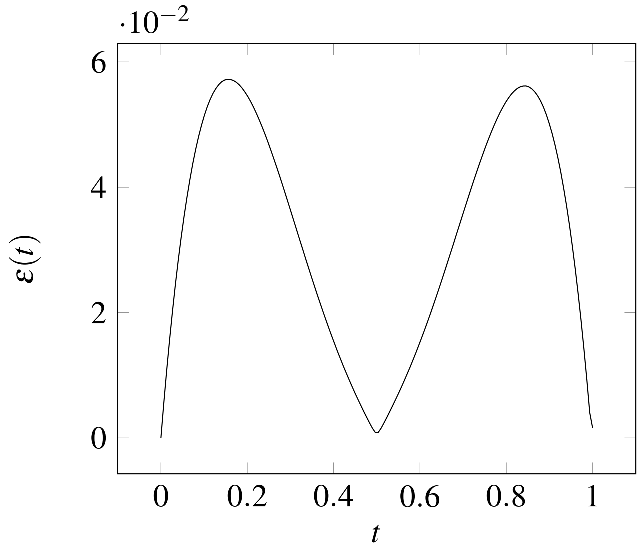

Figure 4.6: Error of the approximation

Fig 4.6 shows that the maximum error of the approximation is 0.0575, which occurs twice at $t = 0.157, t = 0.843$. 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 the Bézier approximation has a horizontal curve, because the control points P_0 and P_1 form a flat horizontal line. This is shown more clearly in Fig. 4.7.

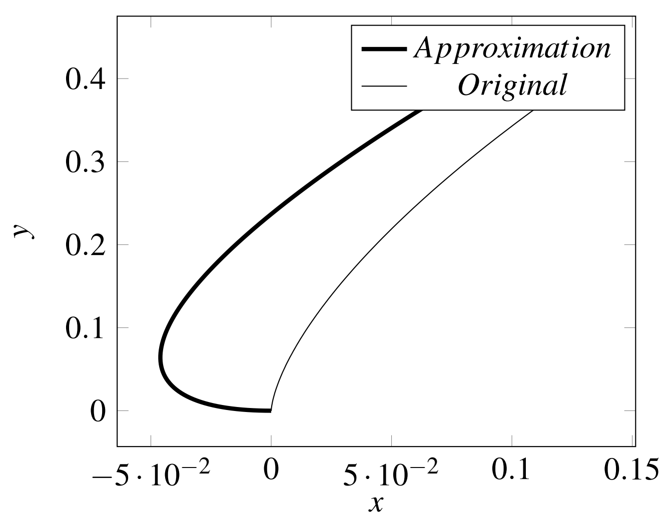

Figure 4.7: Error of the approximation

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

4.3. Approximation Using Iterative Numerical Analysis

With the help of numerical analysis using a computer, a very 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 ($\boldsymbol{P}_0$ and $\boldsymbol{P}_3$, 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 ($\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, 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 is shown in Fig. 4.8.

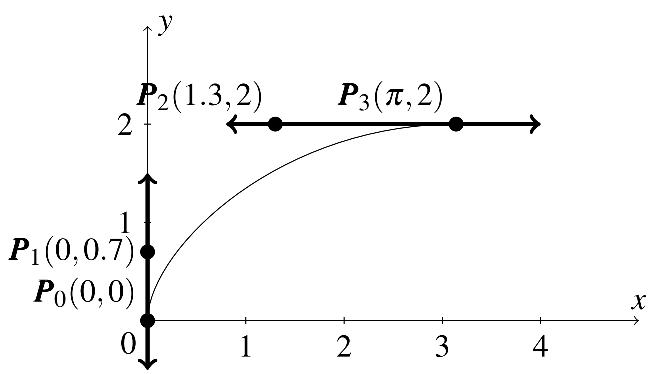

Figure 4.8: Possible arrangement of the control points

As such, the control points will be as following:

$$\boldsymbol{P}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \boldsymbol{P}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ u \end{bmatrix},$$
$$\boldsymbol{P}_2 = \begin{bmatrix} \pi \\ 2 \end{bmatrix} - \begin{bmatrix} v \\ 0 \end{bmatrix}, \qquad \boldsymbol{P}_3 = \begin{bmatrix} \pi \\ 2 \end{bmatrix},$$

where $u, v \in \mathbb{R}$. As such, finding u, v that will minimize $\max\limits_{t\in[0,1]} \varepsilon(t)$ will give a close approximation of a cycloid that fits the boundary conditions.

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

$$\zeta(u,v) = \max_{t\in[0,1]} \varepsilon(t). \tag{11}$$

As such, $\arg\min_{u,v}(\zeta(u,v))$ will yield u,v that will give the best possible approximation. The graph of $\zeta(u,v)$ is plotted using GNU Octave, which is shown in Fig. 4.9. It shows that there is only one minimum of $\zeta(u,v)$ for $u,v \in [-4,4]$.

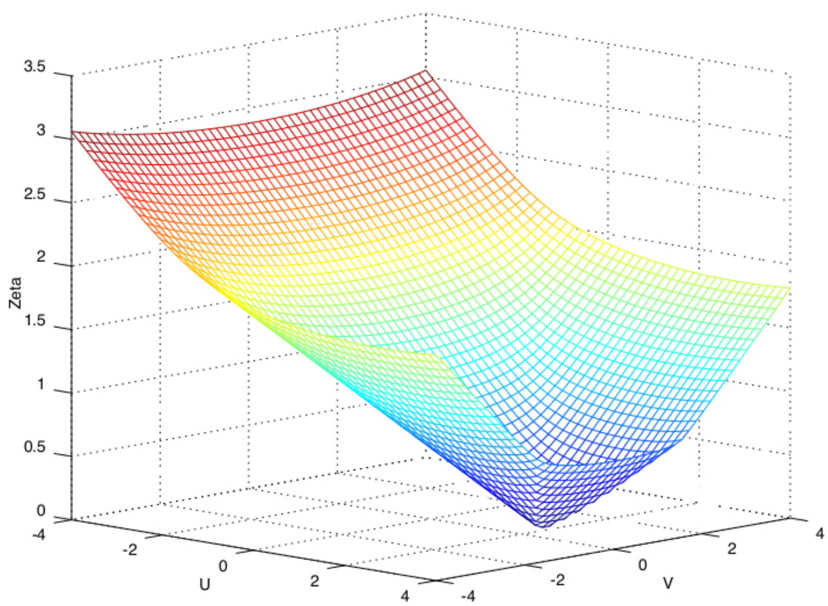

Figure 4.9: Plot of $u,v \in [-4,4]$ and $\zeta(u,v)$

Using GNU Octave, $\arg\min_{u,v}(\zeta(u,v))$ is approximated iteratively, with initial domains of $u \in [-4,4]$ and $v \in [-4,4]$, in which each iteration subdivides the initial interval into 10, finds the minimum, and reiterate the process to find the minimum around the u,v discovered from previous iteration within the radius of previous interval. The 10 subdivision is chosen because it is small enough that the iteration converges into a global minimum within the initial domains. Moreover, it adds approximately 1 significant figure per each iteration. For the specific programming code, see

Appendix A.

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

$$\boldsymbol{P}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \boldsymbol{P}_1 = \begin{bmatrix} 0 \\ 0.000988889 \end{bmatrix},$$
$$\boldsymbol{P}_2 = \begin{bmatrix} \pi - 2.50477 \\ 2 \end{bmatrix}, \qquad \boldsymbol{P}_3 = \begin{bmatrix} \pi \\ 2 \end{bmatrix}.$$

Graphing the Bézier approximation yields the following:

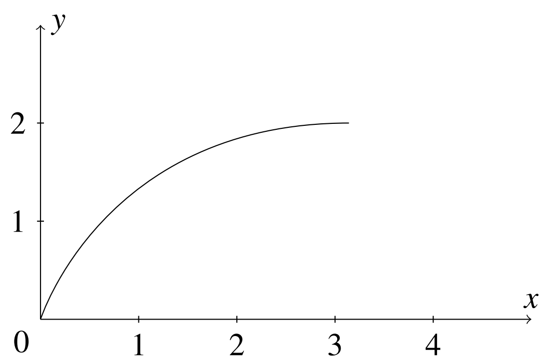

Figure 4.10: Bézier approximation using iterative numerical analysis

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

5. Evaluation and Conclusion

Table 5.1 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 $t=0$	$\max\limits_{t\in[0,1]} \varepsilon(t)$
Maclaurin Polynomial	Yes	Yes	No	No	3.566
Semicircle Approximation by Dokken et al. (1990)	Yes	No	Yes	Yes	0.0575
Iterative Numerical Analysis	Yes	Yes	Yes	Yes	0.0810

Table 5.1: Comparison of Bézier approximations of cycloids

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 satisfying the boundary conditions will not necessarily provide the best approximation. Also, 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 is used. Moreover, the traditional method of using Maclaurin polynomial for approximation of a non-polynomial curve is shown to be not appropriate for Bézier approximants, unless a Bézier curve of much higher degree is used. In conclusion, the Bézier curve is a good approximant of a cycloid.

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.

Appendix A GNU Octave Code for Numerical Analysis

```
function extended_essay()
   [u, v, minZeta] = approximate(-4, 4, -4, 4, 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
```