A Novel Approach to Numerically Solving the Brachistochrone Problem

Owen Morehead

March 15, 2022

Abstract

The brachistochrone problem is the curve of fastest descent of a bead which slides frictionlessly between two points, under the influence of a constant gravity. Finding the brachistochrone involves solving the Euler-Lagrange equations, which take the form of a two-point boundary value ODE. Unfortunately, this ODE has a singularity at the boundary, which is not commonly treatable by most built-in ODE solvers. In this paper, we consider a novel approach to deriving the solution which combines asymptotic analysis and numerical routines in order to avoid this singularity. This method is validated against the well known analytical cycloid solution. Importantly, the numerical solutions are unique for all boundary points, i.e. the optimal curve is always returned. Additionally, this solution method can easily be generalized for more complicated variations of this problem where analytical solutions do not exist, for instance when we include other external forces such as viscous friction in a fluid filled environment.

1 Introduction

Methods of optimization play an integral role in advancing science and technology. Optimization in general allows us to continuously achieve the best and most efficient solution to a defined problem. One of the most important *path* optimization problems is known as the *brachistochrone* problem, and was first hypothesized by Galileo and rediscovered around 1696 by Swiss mathematician Johann Bernoulli. Bernoulli posed this problem to the readers of *Acta Eruditorum* as a simple yet demanding question, given what mathematical techniques were available in the 17th century (Bernoulli, 1969). Bernoulli proposed a solution using an analogy to light and what we now know as Fermat's principle of least time (de Icaza, 1993). This was a clever way to relate this problem to a natural optimization phenomenon, which is that light manages to always travel from one point to another in the shortest possible time. Around the same time, unique solutions to this problem were also proposed by other famous mathameticians such as Isaac Newton, Gottfried Leibniz, and Guillaume de L'Hôpital (de Icaza, 1993).

The word brachistochrone stems from the Ancient Greek words, *Brákhistos khrónos*, meaning 'shortest time'. Thus, the brachistochrone curve is simply the curve of shortest time, or fastest descent, but with slightly more detail to the specific problem. We consider a spherical bead which slides without friction under the influence of a uniform downward gravitational field. The bead lies in the plane between its starting point A, and the lower point B (not directly below A). The question is then, what is the curve this bead should follow to move from A to B in the shortest amount of time? This problem is not only elegant in the simplicity of the question, but also in the many solutions it attracts. The original solution to this problem was crucial in the development of what we now call the calculus of variations, a widely used topic concerned with the extrema of functionals (a function whose input is another function). In essence, the extrema of functionals may be obtained by finding functions for which the functional derivative is equal to zero. This leads to solving the widely used Euler-Lagrange equation which was developed in the 1750's by Leonhard Euler and Joseph-Louis Lagrange in connection with their studies of the brachistochrone problem. These developments were crucial in forming the basis of modern physical and mathematical topics such as optimization theory and its widely used applications.

The goal of this paper is not to analytically derive this problem's solution using the Euler-Lagrange equation, as is the most common method seen today. Rather, we provide a unique hybrid approach that utilizes both analytical and numerical methods to derive the brachistochrone curve and its corresponding time of fastest descent between two given boundary points. This numerical approach, after confirming it returns correct results, will further allow us to analyze how the brachistochrone curve and its time of fastest descent compares to other general curves for varying boundary points. The approach taken in this paper is useful for many reasons. One such being that it can be easily transformed to generate solutions to variants of this problem with additional parameters such as friction, drag, or viscosity. Solving more complicated variations of the brachistochrone problem become much more difficult and even impossible in some cases if an entirely analytical approach is taken.

The following sections will provide a bottoms up methodology to solving the brachistochrone problem. We first utilize fundamental physics and calculus to derive the two point boundary value ordinary differential equation (ODE) which governs the problem. This ODE can also be found using the Euler-Lagrange equation, and we can confirm these results are indeed the same. Next, we show the steps taken in order to ultimately solve this ODE utilizing a numerical solver (such as Matlab's `BVP4C`). We further compare this methods results against the analytically derived solution to evaluate any error built into this methodology. Furthermore, we compare the brachistochrone's travel time to that of other general curves for various boundary points and draw conclusions based on these results. The final section of this paper will summarize and discuss possible applications of this particular solution approach.

2 Model and Method A First Principle Derivation

2.1 Deriving the Functional

Let us formally describe the brachistochrone problem, which goes as follows. Given two point A and B (with A lying higher than B), find the path along which an object would slide (disregarding any friction) in the shortest possible time from A to B, if it starts from rest and is only accelerated by Earth's gravity. Figure 1 illustrates the geometry of this problem, which lies in two-dimensional Euclidean space, $(x, y) \in \mathbb{R}^2$. We consider the object to be a spherical ball, which starts at a general point $A = (0, y_a)$ and ends at a general point $B = (x_b, 0)$. For this coordinate system chosen, we always have $y_a > 0$ and $x_b > 0$. Note that the example curve shown in Figure 1 is not actually the brachistochrone curve, rather a section of the parabola, $y = (1 - x)^2$.

Before we can start looking for the optimal trajectory $y(x)$, we first start by deriving a formula for the travel time of an object from A to B given a general function $y(x)$. We can write the total time as an integral over each incremental time segment. We can further relate the instantaneous time and distance by the velocity, $v = \frac{dr}{dt} \rightarrow dt = \frac{dr}{v}$, where dr is the incremental path length. Thus, we can define the total time, T in integral form:

$$T = \int_0^T dt = \int_A^B \frac{dr}{v}. \tag{1}$$

To determine the instantaneous velocity, v, in Equation 1, we can use conservation of energy, which implies that the sum of the kinetic and potential energy of the object always remains constant. Since the object is at rest at the initial boundary point, $A = (0, y_a)$, it only has potential energy. We can equate this to the sum of the potential and kinetic energy at any point along the objects trajectory:

$$mgy_a = \frac{1}{2}mv^2 + mgy(x) \quad \longrightarrow \quad v = \sqrt{2g(y_a - y(x))} \tag{2}$$

where g is Earth's gravitational acceleration, which we take to be $9.8m/s^2$. We can further simplify Equation 1 by writing the infinitesimal path length in terms of its components (by Pythagoras' theorem), and making simplifications such as:

$$dr = \sqrt{(dx)^2 + (dy)^2} = \sqrt{1 + \frac{(dy)^2}{(dx)^2}}dx = \sqrt{1 + y'^2}dx. \tag{3}$$

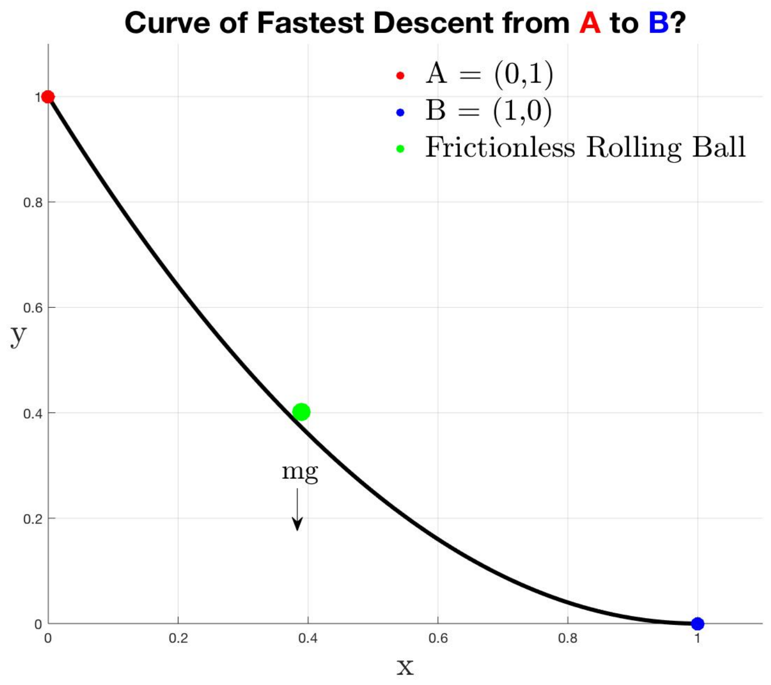

Figure 1: The geometry of the problem. In the x-y plane, the brachistochrone curve is the curve that minimizes the time traveled between points A and B under the influence of a constant downward gravitational field, and ignoring any frictional forces.

Substituting Equation 2 and 3 into Equation 1 now gives us

$$T = \int_{0}^{x_b} \frac{\sqrt{1+y'^2}}{\sqrt{2g(y_A - y(x))}} dx, \tag{4}$$

where we are now integrating from the x-coordinate of A to the x-coordinate of B. We can explicitly see that $T = \mathcal{F}(y(x))$ is indeed a function of $y(x)$. More generally, a large class of functionals can be written in integral form as,

$$\mathcal{F}(y(x)) = \int_{x_1}^{x_2} L(x, y(x), y^{'}(x)) dx \tag{5}$$

where L is twice continuously differentiable. Similar to finding the extrema of functions, the extrema of *functionals* can be obtained by finding a function which results from setting the functional derivative equal to zero. This leads to solving the famous 2-point boundary value problem know as the Euler-Lagrange (EL) equation of the calculus of variations, defined by:

$$\frac{\partial \mathcal{L}}{\partial y} = \frac{d}{dx}(\frac{\partial \mathcal{L}}{\partial y'}). \tag{6}$$

Furthermore, if one is to use the EL equation to optimize our functional described by Equation 4, notice that the integrand does not explicitly depend on the variable of integration, x. If $\frac{\partial L}{\partial x} = 0$, then the EL equation reduces to what is known as the Beltrami identity,

$$L - y^{'} \frac{\partial L}{\partial y'} = \text{Constant}. \tag{7}$$

Using this to solve for the pair of parametric solution equations, $x(C, \theta)$ and $y(C, \theta)$, can be found in the Appendix. These equations describe a segment of what is known as a cycloid. Although the analytical solution is elegant in its entirety, it is limited in terms of solving any more complicated versions of the brachistochrone problem. This method can also be problematic since there are sometimes more than one solution returned for a given choice of boundary points, even though only one of the solutions minimizes the descent time. This is another reason why the unique solutions returned from the numerical method presented below are advantageous in comparison. In what follows, we propose a different route to the derivation of the EL equations which uses only very simple tools of calculus. After all, mathematicians solving this problem in the late 17th century did not have variational calculus in their tool belt like we do now. This method shown below more generally showcases how we can use first principles to optimize any functional, where the function $y(x)$ is fixed on the boundary.

2.2 A First Principles Approach to Optimizing the Functional $T(y(x))$

The general premise of this method is to discretize the curve $y(x)$ so that it is composed of N **linear segments**. Each line segment, connecting the point (x_i, y_i) to (x_{i+1}, y_{i+1}), can be described by the equation:

$$y(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) \qquad \text{for } x \in [x_i, x_{i+1}] \tag{8}$$

where $i = 0, 1, 2, \ldots, N$ is the index of each point connecting the segments, and where $(x_0, y_0) = (0, y_a)$ and $(x_N, y_N) = (x_b, 0)$ are the coordinates for the initial and final points of the curve, A and B. Figure 2 shows the same general curve connecting A and B as in Figure 1, in addition to this piecewise linear approximation of the curve as a sum of four linear segments.

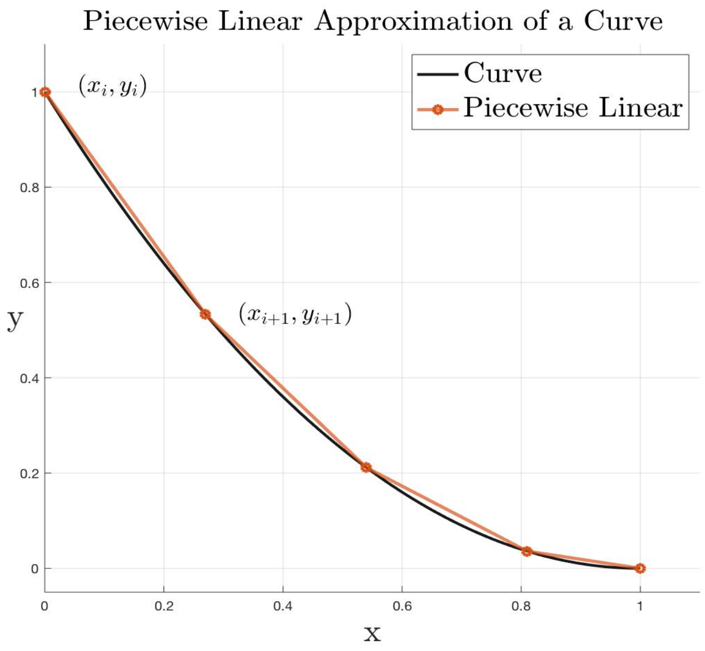

Figure 2: Illustration of the piecewise approximation of $y(x)$ by connected linear segments. The points that connect each line segment are labeled as (x_i, y_i) and (x_{i+1}, y_{i+1}) where $i = 0, 1, 2, \ldots, N$.

We can now approximate the total time T defined by Equation 4 as a sum of integrals over each connecting line segment:

$$T(\{x_i, y_i\}) = \lim_{N\to\infty} \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} \frac{\sqrt{1 + (\frac{y_{i+1}-y_i}{x_{i+1}-x_i})^2}}{\sqrt{2g(y_a - (y_i + \frac{y_{i+1}-y_i}{x_{i+1}-x_i}(x - x_i))}} dx. \tag{9}$$

The integral in Equation 9 can be found analytically, which leads to the simplified form:

$$T(\{x_i, y_i\}) = \lim_{N\to\infty} \sum_{i=0}^{N} \frac{-2}{\sqrt{2g}} \sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2}\left(\frac{\sqrt{y_a - y_{i+1}} - \sqrt{y_a - y_i}}{y_{i+1}-y_i}\right). \tag{10}$$

Note how the functional T is now a function of many variables (x_i and y_i for $i \in [0, \infty]$), so minimizing T consists of finding where the gradient of T with respect to x_i and y_i is equal to zero. Taking the derivative with respect to x_i while holding y_i constant or vise versa has the same effect since the coordinate pair (x_i, y_i) describes one point. Requiring $\frac{\partial T}{\partial x_i} = 0$ is easier since there are fewer terms that involve x. Skipping the intermediate simplification steps, we arrive at the result:

$$\begin{aligned} &\frac{2}{\sqrt{2g}}\left(\frac{\sqrt{y_a - y_{i+1}} - \sqrt{y_a - y_i}}{y_{i+1}-y_i}\right)\left[\frac{x_{i+1}-x_i}{\sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2}}\right] \\ &= \\ &\frac{2}{\sqrt{2g}}\left(\frac{\sqrt{y_a - y_i} - \sqrt{y_a - y_{i-1}}}{y_i - y_{i-1}}\right)\left[\frac{x_i - x_{i-1}}{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}\right]. \end{aligned} \tag{11}$$

In the limit as the total number of points considered, $N \to \infty$,

$$x_{i+1} - x_i \to \Delta x \to 0 \quad , \quad x_i - x_{i-1} \to \Delta x \to 0$$

and similarly for y. Thus we will have that,

$$\frac{\Delta y}{\Delta x} = \frac{dy}{dx} = y^{'}.$$

Noting that both sides of Equation 11 share the exact same form, they can be set equal to some constant, K, and with these two steps, Equation 11 becomes:

$$\frac{y_{i+1}-y_i}{\sqrt{y_a - y_{i+1}} - \sqrt{y_a - y_i}}\sqrt{1+y'^2} = \frac{2}{\sqrt{2g}K} = L \qquad [L = \frac{2}{\sqrt{2g}K}] \tag{12}$$

where we define a new constant $L = 2/(\sqrt{2g}K)$. To simplify further, notice that using the difference of squares allows us to write

$$\frac{y_{i+1}-y_i}{\sqrt{y_a - y_{i+1}} - \sqrt{y_a - y_i}} = \sqrt{y_a - y_{i+1}} + \sqrt{y_a - y_i}. \tag{13}$$

and furthermore, in the limit as $i \to \infty$,

$$\sqrt{y_a - y_{i+1}} + \sqrt{y_a - y_i} \to 2\sqrt{y_a - y_i}.$$

Thus, after squaring both sides, we end up with the two point boundary value ODE whose solution is the optimal trajectory for the brachistochrone problem:

$$(y_a - y)(1 + y^{'2}) = C \qquad [C = \frac{L}{2}] \tag{14}$$

where C is a newly defined constant in terms of L. The associated boundary conditions are,

$$f(x=0) = y_a \quad \text{and} \quad f(x = x_b) = 0.$$

Indeed, if we were to use the Beltrami identity to optimize the functional described by Equation 4, we would arrive at the same results (see Appendix). This is good confirmation that this first principle method returns the same ODE. Although a slightly longer process, this method can be useful when dealing with even more complicated functionals which cannot easily be solved with the EL equation. Any curve can be described as a sum of many linear segments, as per the reasoning behind this approach.

3 Deriving the Solution of the Optimal Trajectory

One might hope that solving this boundary value ODE would be fairly trivial if we use a built-in numerical solver. However this is not quite the case. Notice that as $(x, y) \to (x_a, y_a)$, the denominator of the ODE rewritten as,

$$(1 + y^{'2}) = \frac{C}{(y_a - y)} \tag{15}$$

tends to zero. In other words, this ODE is singular at the A boundary point. Furthermore, this type of singularity is not the kind that is commonly treatable by most built-in ODE solvers since it is singular in y and not x. In short, standard numerical integration techniques fail at this boundary point. Thinking back to the problem at hand, the ball is starting from rest and hence it takes a significant amount of time to build acceleration in the vicinity of $x \approx x_a$. It is therefore necessary to include this region in an accurate total time calculation.

In what follows, we make use of asymptotic theory to solve for $y(x)$ near $x \approx x_a$ analytically, and thus estimate its contribution to $T(y(x))$. To do so, we split the domain $[x_a, x_b]$ into two:

$$l_\epsilon = [0, \epsilon] \qquad \text{and} \qquad l_{\text{bulk}} = [\epsilon, x_b]$$

where ϵ represents a tiny shift away from the singular boundary, $|\epsilon| \ll 1$.
This means the integral T now looks like,

$$T = \frac{1}{\sqrt{2g}} \int_0^\epsilon \frac{\sqrt{1 + y'^2(x)}dx}{\sqrt{(y_A - y(x))}} + \frac{1}{\sqrt{2g}} \int_\epsilon^{x_B} \frac{\sqrt{1 + y'^2(x)}dx}{\sqrt{(y_A - y(x))}}. \tag{16}$$

The three main steps to this solution process follow:

1. Numerically solve the ODE described by Equation 15 for $x \in [\epsilon, x_b]$ with a new boundary condition to be determined. There is no singularity in this domain so any numerical ODE solver will now work.
2. Analytically solve the first integral in T (Equation 16) once we find equations for $y^{'}(x)$ and $y(x)$ in $x \in [0, \epsilon]$, to be determined.
3. Solve the second integral in T (Equation 16) via numerical integration since we have numerically found $y^{'}(x)$ and $y(x)$ for $x \in [\epsilon, x_b]$.

We start with **step 1**. The first thing to notice is that we can make the approximation,

$$1 + y^{'2} \approx y^{'2} \quad \text{near} \quad y_a.$$

This is because y is singular at y_a, i.e. $y^{'2} \gg 1$, and adding 1 to a value near infinity will not have any significant affect. We can see that near y_a, the right hand side denominator of Equation 15 will be very small and hence $1 + y^{'2} \to \infty$. Therefore we have the approximated first order ODE, which we can separate variables and easily integrate:

$$y^{'2} = \frac{C}{(y_a - y)} \quad \to \quad \frac{dy}{dx} = \pm\sqrt{\frac{C}{(y_a - y)}} \quad \to \quad \int \sqrt{y_a - y}dy = \int \pm\sqrt{C}dx \tag{17}$$

which we get,

$$-\frac{2}{3}(y_a - y)^{\frac{3}{2}} = \pm\sqrt{C}x + C \tag{18}$$

leading to a solution in terms of $y(x)$,

$$(y_a - y) = (\frac{3}{2}\sqrt{C}x)^{2/3} \tag{19}$$

and a derivative,

$$(y_a - y)^{'} = (-y)^{'} = (\frac{3}{2}\sqrt{C})^{2/3}(\frac{2}{3}x^{-1/3}) \tag{20}$$

where we found that the integration constant is zero due to the boundary condition at A. Also notice that at any point along the curves trajectory, $y \leq y_a$. Therefore, for both sides of Equations 18 to share the same sign, we must only consider the negative right hand side solutions. Furthermore, we eliminate the constant C by dividing Equation 20 by Equation 19, leaving us with,

$$\frac{-y^{'}}{(y_a - y)} = \frac{2}{3}x^{-1} \tag{21}$$

which now leaves us with the new boundary condition to use at $x = \epsilon$, namely,

$$y^{'}(\epsilon) + \frac{2}{3}\epsilon^{-1}(y_a - y(\epsilon)) = 0. \tag{22}$$

To numerically solve the 2nd order ODE in the domain $x \in [\epsilon, x_b]$, the built-in function chosen was Matlab's `BVP4C` fourth-order boundary value problem solver. We can take a derivative of the first order ODE in Equation 14 to arrive at,

$$y^{''} = \frac{1 + y^{'2}}{2(y_a - y)} \tag{23}$$

which can now can be rewritten as a system of two first-order equations:

$$y_1^{'} = y_2 \tag{24}$$

$$y_2^{'} = \frac{1 + y^{'2}}{2(y_a - y)} \tag{25}$$

with corresponding boundary conditions:

$$y_2(\epsilon) + \frac{2}{3}\epsilon^{-1}(y_a - y_1(\epsilon)) = 0 \tag{26}$$

$$y_1(b) = 0. \tag{27}$$

Using the `BVP4C` solver with an additional initial solution guess (e.g. linear) outputs discrete solution values for $y_1(x) = y(x)$ and $y_2(x) = y^{'}(x)$.

Moving on to **step 2**, we can now analytically solve for the total time in the domain $x \in [0, \epsilon]$. Again, we note that in this domain, $1 + y^{'2} \approx y^{'2}$ and thus,

$$T \approx \frac{1}{\sqrt{2g}} \int_0^{\epsilon} \frac{\sqrt{y'^2(x)}dx}{\sqrt{(y_A - y(x))}} = \frac{1}{\sqrt{2g}} \int_0^{\epsilon} \frac{(\frac{3}{2}\sqrt{C})^{2/3}(\frac{2}{3}x^{-1/3})}{(\frac{3}{2}\sqrt{C}x)^{1/3}}dx \tag{28}$$

using Equations 19 and 20. Solving this integral yields,

$$T = \sqrt{\frac{2}{g}}\Big(\frac{3}{2}\Big)^{1/3} C^{1/6}\epsilon^{1/3} \tag{29}$$

with C found from Equation 19 evaluated at ϵ,

$$C = \Big(\frac{y_a - y(\epsilon)}{(\frac{3}{2}\epsilon)^{2/3}}\Big)^3 \tag{30}$$

where $y(\epsilon)$ is obtained from the numerical solution in l_{bulk}. We now have a quantity describing the total descent time from $x = 0$ to $x = \epsilon$.

In **step 3**, finally, we compute the integral in l_{bulk} numerically. We can simply use the resulting discrete solutions values for $y^{'}(x)$ and $y(x)$ from `BVP4C` to numerically solve the integral,

$$T = \frac{1}{\sqrt{2g}} \int_{\epsilon}^{x_B} \frac{\sqrt{1 + y'^2(x)}dx}{\sqrt{(y_A - y(x))}} \tag{31}$$

using a built-in function such at Matlab's trapezoidal integration, `trapz()`. We are now left with the desired result. We can easily plot any brachistochrone curve, and calculate the shortest time it takes the ball to roll from any general point $A = (0, y_a)$ to $B = (x_b, 0)$ by summing together the integral values obtained from steps 2 and 3.

4 Method Validation

In this section we first test our numerical algorithm against the well known analytical solution. As described in the above section, the numerical method uses a variable ϵ which is a very small shift away from the origin in order to avoid the singularity at the $A = (0, y_a)$ boundary point. Since the solution in l_ϵ is calculated using asymptotic methods, we expect the results to depend weakly on the choice of ϵ. A first important step in validating our model is to see which value of ϵ leads to the least amount of error in comparison to the analytically derived descent time. Figure 3 shows the error between the numerical and analytical solution for a reasonable domain of ϵ values, the error computed as,

$$\frac{|T_{num} - T_{analytic}|}{T_{num}}.$$

We plot this error for a fixed endpoint, $B = (1, 0)$ and a range of starting points from $A = (2, 0)$ to $A = (10, 0)$. Figure 3 clearly reveals that the error has a minimum around $\epsilon_{\text{best}} = 1 \times 10^{-3}$ for all boundary points. Although the error tends to increase as the starting point moves closer to $y = 0$, we see that even the largest error for an $\epsilon = 1 \times 10^{-3}$ is only 7.48×10^{-5} (corresponding with the starting point $A = (0, 2)$). This demonstrates that these numerical and analytical solutions are equivalent up to intrinsic numerical errors. For $\epsilon < \epsilon_{\text{best}}$, we see the error becomes increasingly large as $\epsilon \to 0$, likely due to the singularity at $y = y_a$. Note that if the numerical solver uses an ϵ close enough to 0 (e.g. $\epsilon = 1 \times 10^{-6}$), it will return a singularity error flag. On the other hand, we see that the error also increases linearly for $\epsilon > \epsilon_{\text{min}}$, but at a slower rate. We hypothesize this error is primarily a result of the total time calculation over the l_ϵ domain, using `Matlab`'s `trapz()` function. In addition, as ϵ increases, the models approximation becomes gradually less accurate. The approximation being that $1 + y'^2 \approx 1$ for $x \approx x_a$. Thus for larger values of ϵ, we expect there to be a slightly larger discrepancy between our numerical and the analytical solutions. Despite this, all error values are reasonable even for larger values of ϵ.

In light of this discussion, we choose $\epsilon = 1 \times 10^{-3}$ for all further numerical analysis. It is also worth noting that the number of grid points used in the numerical solver has an effect on the results as well. Although increasing the amount of grid points will increase computational time, it was found that all calculations are sufficiently accurate for any number of points greater than around 500. Thus we use 500 grid points for all further simulations.

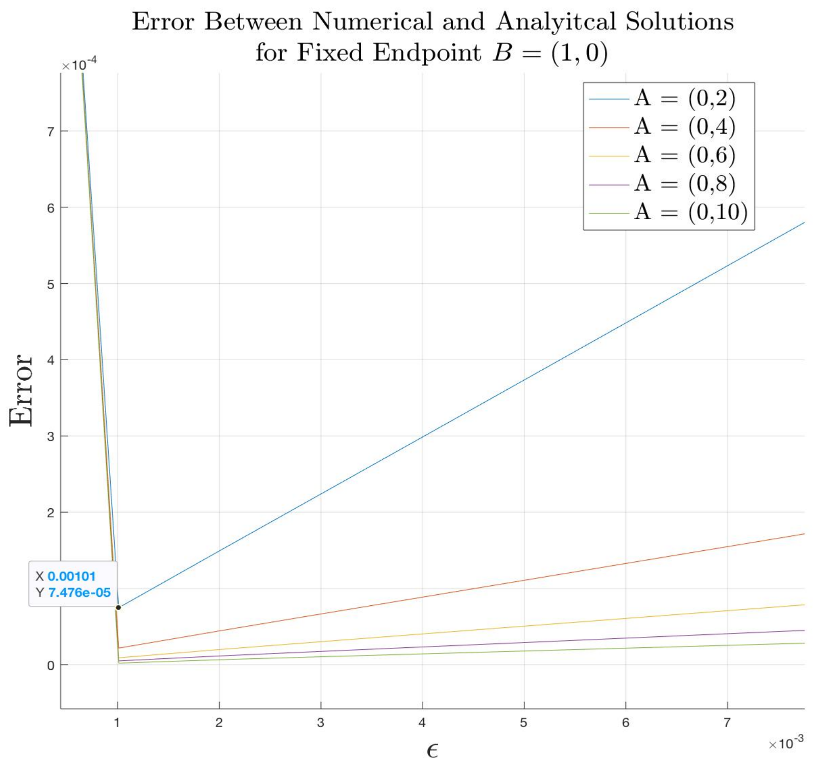

Figure 3: Error between numerical and analytical solutions against ϵ ranging from $\epsilon = 5 \times 10^{-4}$ to $\epsilon = 8 \times 10^{-3}$ in the numerical solution. These calculations are done for a fixed endpoint $B = (1, 0)$ and starting points A ranging from $(2, 0)$ to $(10, 0)$ (see legend).

5 Numerical Results

Now that we have the ability to numerically generate an accurate brachistochrone curve and the total descent time, we begin by visualizing its shape for varying boundary points. Figure 4 shows brachistochrone curves for a fixed starting point $A = (0, 1)$ and varying endpoints $B = (1, 0)$ to $B = (10, 0)$, with respective total times shown in Table 1. We notice the increasingly parabolic shape of the curve as we 'stretch' out the distance the ball travels in the horizontal direction. As x_b increases, the curve must dip further below the x-axis in order to generate enough speed. It is interesting to note that despite the ball's final vertical climb being as large as ≈ 2 meters for an endpoint $B = (10, 0)$, this curve is still faster than any curve decreasing monotonically from $A = (0, 1)$ to $B = (10, 0)$.

In addition, Figure 5 shows brachistochrone curves for a fixed end point $B = (1, 0)$ and varying starting points $A = (x = 0, y = 1, 2, \dots, 10)$, with respective total times shown in Table 2. As expected, the more vertical distance between endpoints, the more linear the curve of fastest descent becomes. After all, the curve of fastest descent from a point A to a point B directly below is a vertical line. Thus, we expect total travel times for a ball that travels increasingly more vertical (Fig 5) compared to horizontal (Fig 4) to be much shorter.

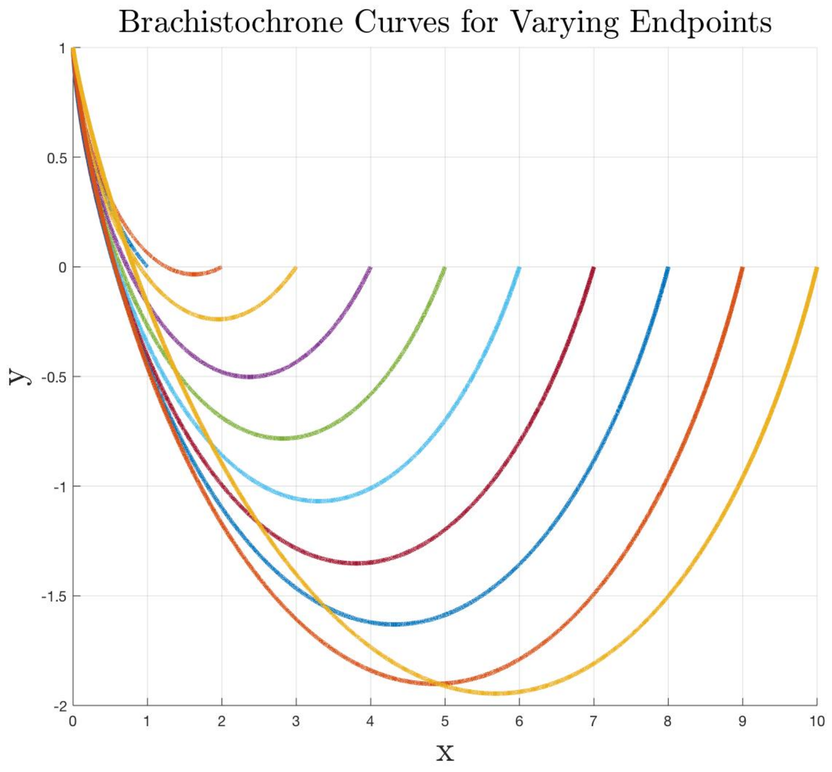

Figure 4: brachistochrone curves for a fixed starting point $A = (0, 1)$ and varying endpoints $B = (1, 0)$ to $B = (10, 0)$.

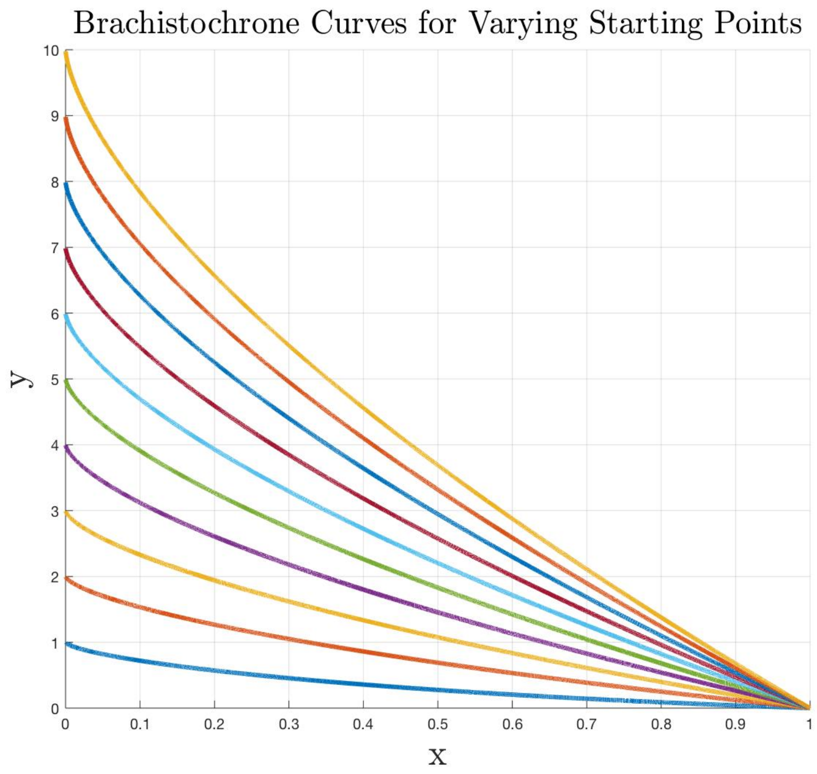

Figure 5: brachistochrone curves for a fixed end point $B = (1, 0)$ and varying starting points from $A = (0, 1)$ to $A = (0, 10)$.

x_b	Total Time (s)
1	0.5831
2	0.8064
3	1.0198
4	1.2138
5	1.3908
6	1.5536
7	1.7033
8	1.8419
9	1.9805
10	2.1533

Table 1: Total descent times corresponding to brachistochrone curves in Figure 4

y_a	Total Time (s)
1	0.5831
2	0.6944
3	0.8148
4	0.9244
5	1.0253
6	1.1182
7	1.2046
8	1.2855
9	1.3618
10	1.4342

Table 2: Total descent times corresponding to brachistochrone curves in Figure 5

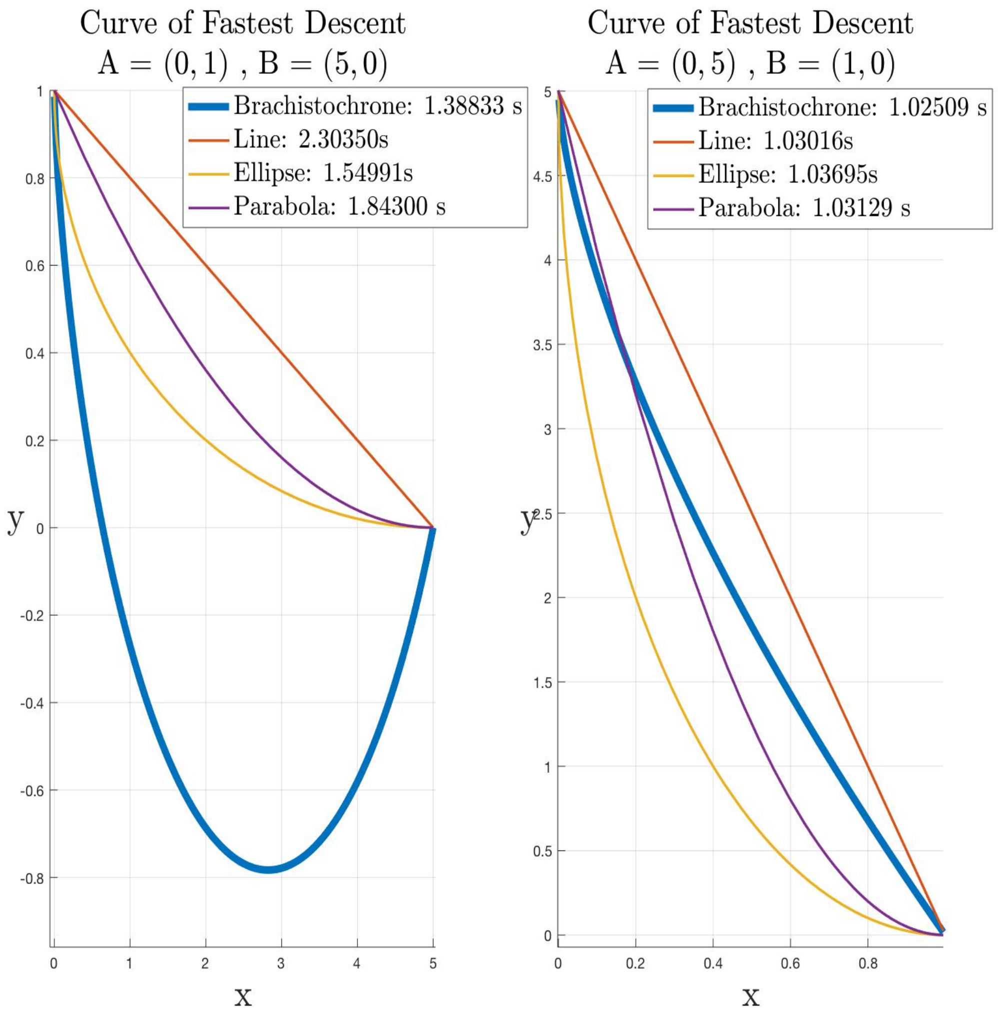

Figure 6: A comparison of the numerically generated brachistochrone curve against other general monotonically decreasing curves for two different sets of endpoints. The legend labels each curve as well as their calculated total descent time. The brachistochrone curve shows is always the curve of fastest descent.

It is important to verify that the brachistochrone curve found is indeed faster than any other general functions such as a line, parabola, section of and ellipse, etc. This provides additional confirmation that our numerical routine is correct. A plot of these curves and the brachistochrone curve between two pairs of boundary points, $A = (0, 1)$, $B = (5, 0)$, and $A = (0, 5)$, $B = (1, 0)$ can be seen above in Figure 6. The equations used for each curve are:

$$\begin{aligned} &\text{Line:} && y = y_a - \frac{y_a}{x_b}x \\ &\text{Ellipse:} && y = y_a\left[1 - \left(1 - (\frac{x}{x_b} - 1)^2\right)^{1/2}\right] \\ &\text{Parabola:} && y = \frac{y_a}{x_b^2}(x - x_b)^2. \end{aligned} \tag{32}$$

Again, these simulations were done with a value of $\epsilon = 1 \times 10^{-3}$. We see the general monotonically decreasing curves all return larger descent times, and the brachistochrone curve is increasingly efficient for an increased x_b. One might suspect that a curve which is not monotonically decreasing would return a more similar descent time to the brachistochrone between the boundary points $A = (0, 1)$ and $B = (5, 0)$, since the brachistochrone curve is also of this form. Therefore, in Figure 7, we test against the non-monotonic parabola defined by the following equation:

$$\text{Parabola[2] :} \qquad y = \frac{(x-3)^2 - 4}{5}. \tag{33}$$

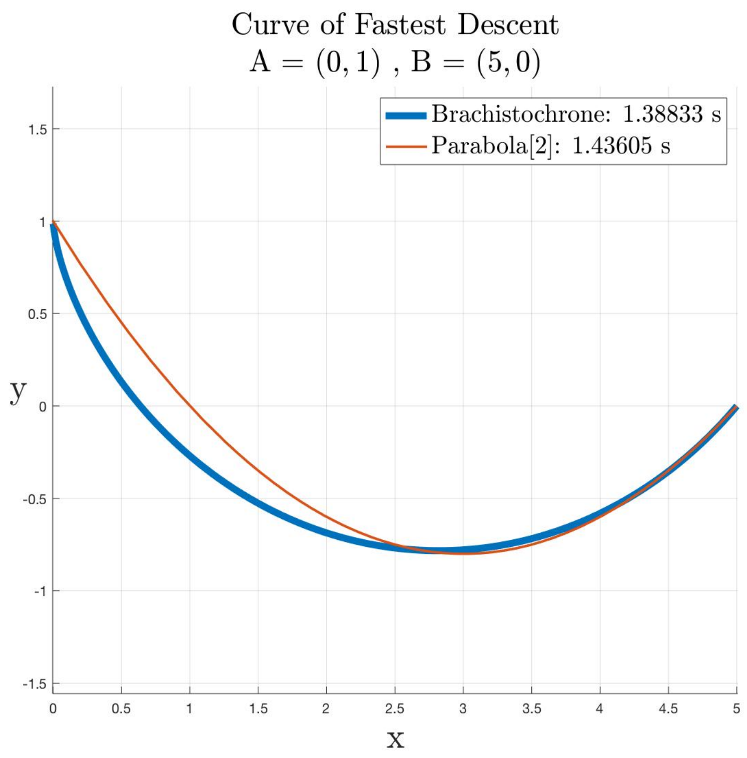

Figure 7: A comparison of the numerically generated brachistochrone curve against a non-monotonic parabola defined by Equation 32. The legend labels each curve as well as their total descent time.

We see that both curves are now similar in shape, and as a result share similar descent times. The fact that the parabola dips below the x-axis and forms a local minimum allows it to generate more speed compared to any monotonically decreasing curve in Figure 6. However, it is still slightly slower than the brachistochrone as expected. The near-vertical initial trajectory of the brachistochrone curve accelerates the rolling ball much faster than any general curve without this feature. In comparison to the case where the ball primarily travels vertically downwards (Figure 6, right), resulting descent times are now approximately equal for any of the curves. In this scenario, one would expect any non-monotonic curve which dips below the x-axis to be slower than one which does not, since the ball already generates enough speed in its direct trajectory towards x_b. We can further visualize how the total descent times change for each curve, first as a function of varying x_b (Figure 8) and similarly as a function of varying y_a (Figure 9).

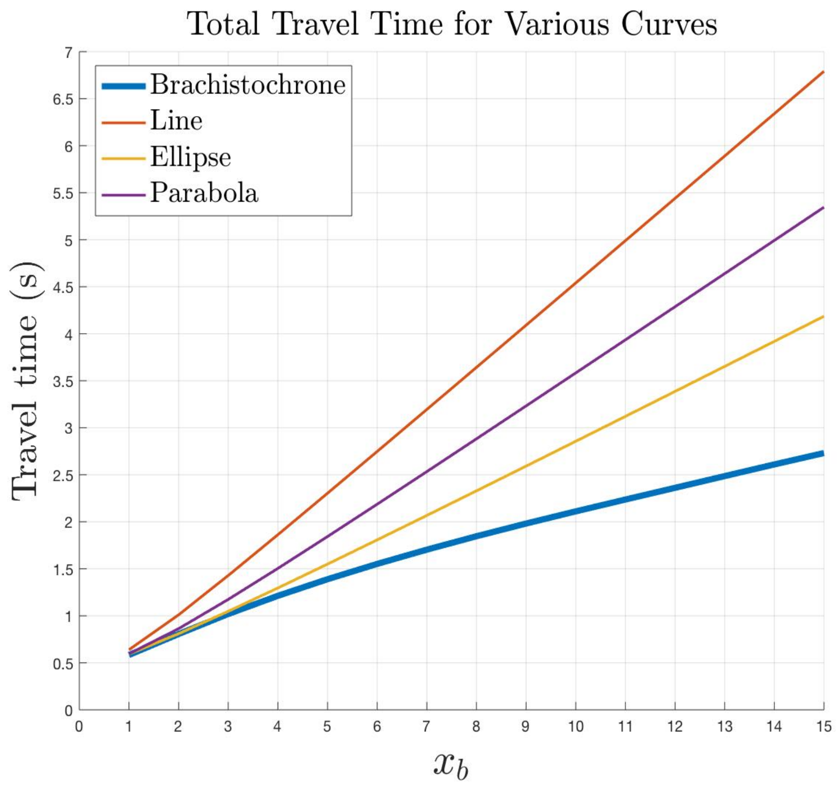

Figure 8: Total time recorded from the numerically generated brachistochrone curve against other general curves (line, ellipse, parabola). All curves start at $A = (0, 1)$ and the endpoints are varied from $B = (1, 0)$ to $B = 15, 0)$.

We can clearly see how descent times of the different curves start to deviate as the boundary points change. Indeed, the brachistochrone curve's descent time remains the fastest over all varying x_b and y_a. This is much more evident in Figure 8. The brachistochrone is increasingly efficient in comparison with the other curves when x_b increases, because all the others are monotonic while it is not. Where, say, the total travel time for a line is only 9% slower compared to a brachistochrone for boundary points $A = (0, 1)$ and $B = (1, 0)$, travel time along the line is now around 150% or 2.5 times greater than the brachistochrone curve for a much longer horizontal travel ($A = (0, 1)$ and $B = (15, 0)$). The second fastest descent time comes from the curve of an ellipse, however as x_b increases it still becomes significantly slower than the brachistochrone.

On the other hand, if we increase the starting point further up the y-axis, the difference in travel times between the brachistochrone and the other curves is much smaller (Figure 9). Visually they all seem to have about the exact same travel time for larger values of y_a. This is consistent with the discussion regarding Figures 5 and 6, since the optimal curve is becoming increasingly linear as we increase y_a. However, it is not necessarily obvious that the parabola and ellipse should return such similar descent times, since these curves have notably different shapes. Regardless, we can conclude from these results that the exact shape of the curve matters much less when the ball essentially only needs to travel straight downwards. Any steep and monotonic curve connecting these two endpoints will return a fairly similar descent time.

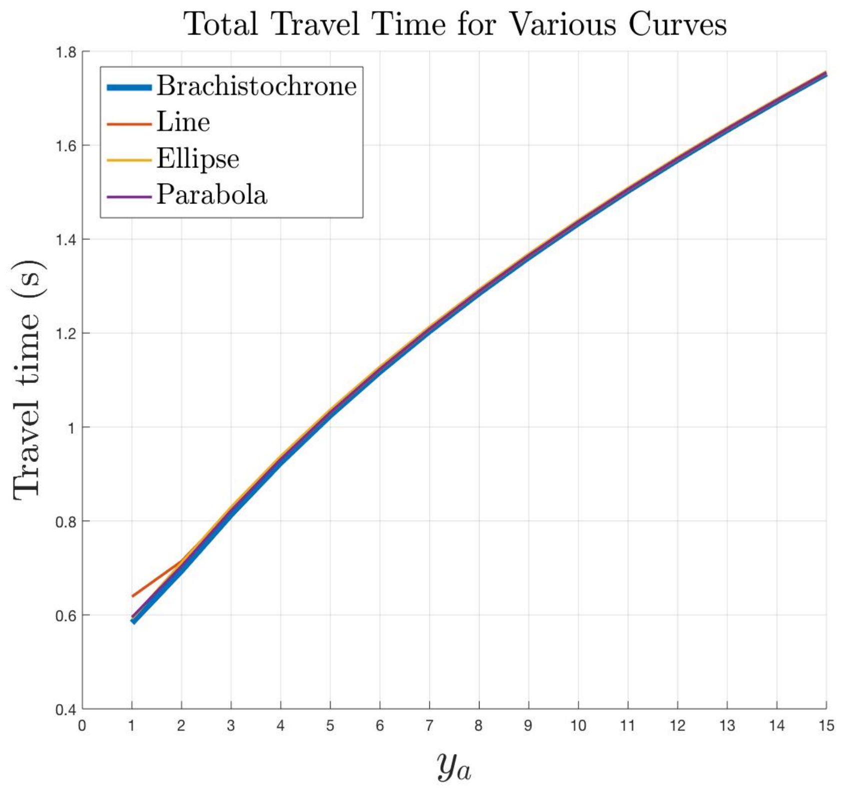

Figure 9: Total time recorded from the numerically generated brachistochrone curve against other general curves (line, ellipse, parabola). All curves end at $B = (1, 0)$ and the starting point is varied from $A = (0, 1)$ to $A = (0, 15)$.

6 Conclusion

Although a seemingly simple question, deriving the brachistochrone curve and its total descent time from point A to B is not necessarily trivial when done numerically. This unique approach started with basic calculus principle to showcase how to optimize the general type of functional which this problem entails. This led us to a governing ODE to the brachistochrone problem, which happens to be singular on the boundary. Without the singularity, this problem would be easy to solve numerically. However, the presence of the singularity precludes the use of "standard" algorithms, so we solved the problem by splitting the domain into $[0, \epsilon]$ and $[\epsilon, x_b]$ and using asymptotic theory in the first of these intervals. After performing these steps, an overarching goal of this paper was to confirm this methodology does indeed return the correct results.

A comparison with the analytically derived descent times confirms this method works. We find that the difference between analytical and numerical results is small for various choices of numerical integration parameters (ϵ and the total grid size). Furthermore, a value of $\epsilon = 1 \times 10^{-3}$ seemed to generate the smallest error between numerical and analytical results. The numerical error varies with the choice of x_b and y_a, however at $\epsilon = 1 \times 10^{-3}$, all corresponding error values were the smallest. We therefore consider this to be the best value to use in any further implementations of this specific model.

All in all, this numerical approach has enabled us explore further than most standard analytical methods would have taken us; visualizing the brachistochrone curve and its fastest descent time for varying starting and endpoint boundaries. The parameterized analytical solution shown in the Appendix can be problematic because there are often more than one solution returned for a given choice of boundary points A and B, even though only one of these minimizes the descent time. This limits the amount of analysis that can be done without taking the extra step to confirm which solution is correct. The numerical method, on the other hand, does not suffer from this arbitrariness, and always returns the optimal curve between any A and B fixed on the boundary.

This same methodology can easily be generalized to explore variations of this simple brachistochrone problem, where analytical solutions may not exist. These problem variations might involve more realistic features such as friction, viscosity, or any arbitrary gravity field. For example, it would be necessary to consider viscous frictional forces if this same problem took place in some fluid filled environment. Indeed, research in this area is being explored currently and has many possible implementations (Gurram et al, 2019). Optimizing for the shortest travel time of an object in an environment underwater, or in space (where both gravity and the atmosphere can be vastly different from Earths) are unique problems which the methodology proposed in this paper could help explore.

References

de Icaza Herrera, Miguel (1993), ”Galileo, Bernoulli, Leibniz and Newton around the brachistochrone problem.” Revista Mexicana de Física 40.3: 459-475.

Johann Bernoulli (June 1696) ”Problema novum ad cujus solutionem Mathematici invitantur.” (A new problem to whose solution mathematicians are invited.), Acta Eruditorum, 18 : 269. From p. 269:

Parnovsky, A. S. (1998), ”Some generalisations of brachistochrone problem.” Acta Physica Polonica- Series A General Physics 93.145: 55-64.

Appendix

An analytical solution to the brachistochrone problem using the Euler-Lagrange equation and further the Beltrami identity. Remember a functional of the form,

$$\frac{\partial \mathcal{L}}{\partial y} = \frac{d}{dx}(\frac{\partial \mathcal{L}}{\partial y^{'}}). \tag{34}$$

can be solved using the Beltrami identity if it is not a function of x.

$$L - y^{'}\frac{\partial L}{\partial y^{'}} = C \qquad\qquad (C = \text{constant}). \tag{35}$$

Using the functional we found in Equation 4 we have,

$$\frac{\sqrt{1+y'^2}}{\sqrt{2g(y_A - y)}} - \frac{y^{'2}}{\sqrt{2g(y_A - y)(1+y'^2)}} = C \tag{36}$$

$$C\sqrt{2g(y_a - y)(1+y'^2)} = 1 \tag{37}$$

$$\sqrt{(y_a - y)(1+y'^2)} = K \qquad\qquad [K = \frac{1}{C\sqrt{2g}}] \tag{38}$$

$$y^{'} = \frac{dy}{dx} = \sqrt{\frac{D-(y_a - y)}{(y_a - y)}} \qquad\qquad [D = K^2] \tag{39}$$

$$dx = dy\sqrt{\frac{(y_a - y)}{D-(y_a - y)}} \tag{40}$$

Now we have a form which we can integrate both sides and solve for x,

$$x = \int\sqrt{\frac{(y_a - y)}{D-(y_a - y)}}dy. \tag{41}$$

This integral is solved by parametric equations. We first start by parameterizing y as,

$$y = y_a - D\sin^2(\frac{\theta}{2}) \tag{42}$$

$$dy = -D\sin(\frac{\theta}{2})\cos(\frac{\theta}{2})d\theta. \tag{43}$$

Plugging this back into our integral we have,

$$x = \int\sqrt{\frac{D\sin^2(\frac{\theta}{2})}{D - D\sin^2(\frac{\theta}{2})}} - D\sin(\frac{\theta}{2})\cos(\frac{\theta}{2})d\theta \tag{44}$$

$$x = -D\int\sin^2(\frac{\theta}{2})d\theta \tag{45}$$

$$x = -D\int(1-\cos\theta)d\theta \tag{46}$$

$$x = -\frac{D}{2}(\theta - \sin\theta) + C \tag{47}$$

where C is a new integration constant. The solution as the following pair of parametric equations can now be written as:

$$x = -\frac{D}{2}(\theta - \sin\theta) + C \qquad\qquad y = y_a - \frac{D}{2}(1-\cos\theta). \tag{48}$$

Using the boundary condition will allow us to zero out C. At $A = (0, y_a)$ we have that $\theta_a = 0$, implying

$$x(0) = C = 0. \tag{49}$$

We are now left with,

$$x = -\frac{D}{2}(\theta - \sin\theta) \qquad\qquad y = y_a - \frac{D}{2}(1 - \cos\theta). \tag{50}$$

The total travel time can now be calculated as:

$$T = \sqrt{\frac{1}{2g}} \int_0^{\theta_b} \sqrt{\frac{1 + (\frac{dy/d\theta}{dx/d\theta})^2}{y_a - y}} \frac{dx}{d\theta} d\theta \tag{51}$$

$$T = \sqrt{\frac{1}{2g}} \int_0^{\theta_b} \sqrt{\frac{1 + \frac{\sin^2\theta}{(1-\cos\theta)^2}}{\frac{D}{2}(1 - \cos\theta)}} \frac{-D}{2}(1 - \cos\theta) d\theta \tag{52}$$

$$T = -\sqrt{\frac{D}{4g}} \int_0^{\theta_b} \sqrt{\frac{(1 - \cos\theta)^2 + \sin^2\theta}{1 - \cos\theta}} d\theta \tag{53}$$

$$T = -\sqrt{\frac{D}{2g}} \int_0^{\theta_b} \sqrt{\frac{1 - \cos\theta}{1 - \cos\theta}} d\theta \tag{54}$$

$$T = -\sqrt{\frac{D}{2g}} \theta_b. \tag{55}$$