Boundary value ODE Using Interval Arithmetic

Under the supervision of
Prof. Shaunak Sen

Yash Kumar
Lokendra Singh Gohil

Problem Statement

- This project aims to solve a Boundary Value problem which involves non-linear differential equations subject to endpoint constraints.
- We aim to solve the problem using techniques like Interval Arithmetic which ensures rigorous bounds.
- Specifically, we are solving the Brachistochrone Problem. For this problem, traditional analytical solutions already exist, but we are trying to solve it using Taylor Series and Interval Arithmetic methods.

Introduction to Brachistochrone Problem

- The Brachistochrone problem is a classical problem in calculus of variations
- Find the shape of the curve down which a bead sliding from rest and accelerated by gravity will slip (without friction) from one point to another in the least time. The term derives from the Greek (brachistos) "the shortest" and (chronos) "time, delay."

Existing Analytical Solution

The time to traverse a segment of the curve is given by:

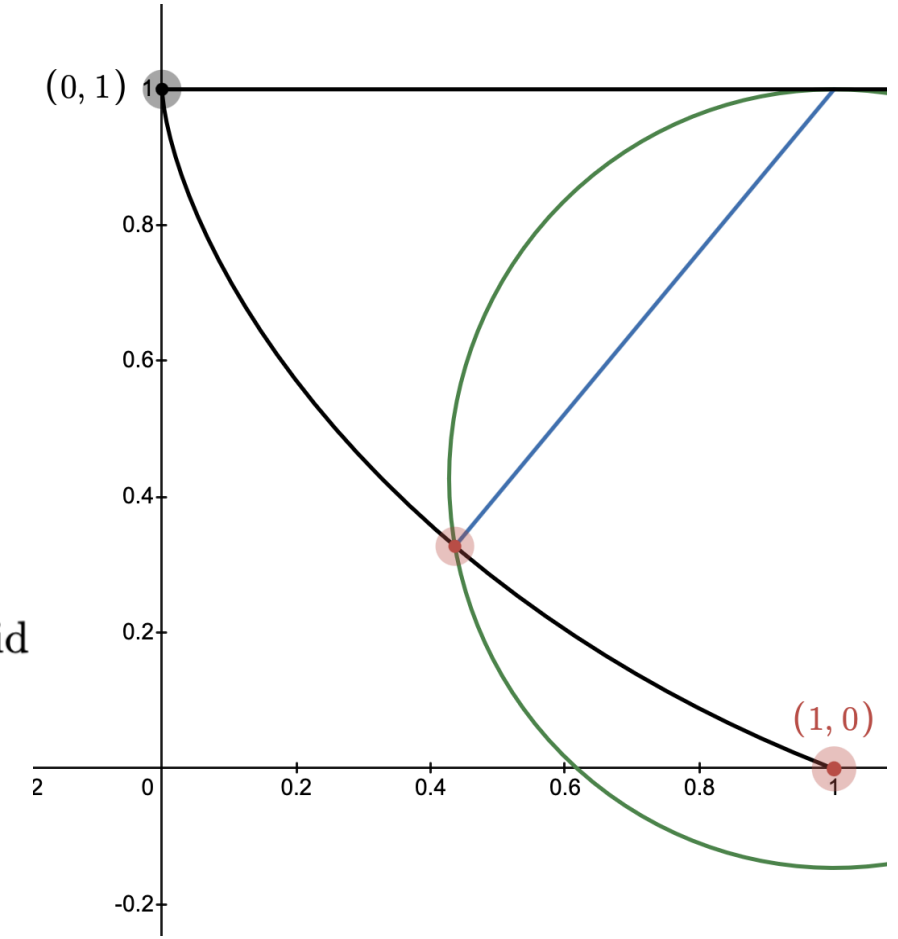
$$T[y, y'] = \int_A^B \frac{\sqrt{1 + \left(\frac{dy}{dx}\right)^2}}{\sqrt{2gy}}\, dx$$

This equation may be simplified to:

$$1 + (y')^2 - 2(y_B - y)y'' = 0$$

The solution curve of the equation, given as parametric equations of cycloid

$$x(\phi) = c_1(\phi - \sin\phi) + c_2,$$
$$y(\phi) = y_B - c_1(1 - \cos\phi)$$



Motivation for using Interval Arthmetics

- Computers work in discrete steps and use floating-point numbers, which can cause rounding errors. For functions with high frquencies these rounding blocks can cause large variations in the calculated values, so we use *interval arithmetic* to track possible ranges of values and keep the results accurate.
- When we use Taylor series, we get polynomial equations. These can be solved accurately using interval version of Newton-Raphson method, guaranteeing solution bounds
- Intervals appear naturally in math. For example, in the Mean Value Theorem:

  $f(\hat{x}) = f(\tilde{x}) + (\hat{x} - \tilde{x})\, f'(\varepsilon)$ with $\hat{x} \le \varepsilon \le \tilde{x}$ cannot be treated numerically in real numbers because ε is unknown. But, the relation $f(\hat{x}) \in f(\tilde{x}) + (\hat{x} - \tilde{x})\, f'(X)$ used numerically $X := [\hat{x}, \tilde{x}]$ with the aid of Interval analysis.

Overview of Our Solution

(Using Taylor Series & Interval Arithmetic)

$$1 + (y')^2 - 2(y_B - y)y'' = 0$$

- The Taylor series expansion is used to approximate the solution to the nonlinear boundary value ODE, providing a systematic way to construct accurate power series representations of the solution over intervals
- The series is substituted into our differential equation to obtain simultaneous equations by coefficient matching
- Interval Newton Method to be used for finding rigorous bounds of the coefficients of the Power series, and the curve represented as an Interval valued function to ensure the solution is enclosed
- By applying interval arithmetic within the boundary value ODE framework, the solution aims to precisely capture uncertainties and verify solution correctness

Taylor Series & Interval Arithmetic Solution

Solving for Initial case

- **The Differential Equation:**

$$1 + (y')^2 - 2(y_B - y)y'' = 0$$

- **The Conditions:**

$$y_B = 1 \qquad (i)\ y(0) = 1 \qquad (ii)\ y(1) = 0$$

- **Taylor Series:**

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots$$

$$y' = a_1 + 2a_2 x + 3a_3 x^2 + \dots$$

$$y'' = 2a_2 + 6a_3 x + \dots$$

- **Expression obtained after substitution:**

$$1 + \left(a_1 + 2a_2 x + 3x^2 a_3\right)^2 - 2\left(2a_2 + 6a_3 x\right)\left(1 - a_0 - a_1 x - x^2 a_2 - x^3 a_3\right) = 0$$

Taylor Series & Interval Arithmetic Solution

Equations obtained on matching coefficients:

$$1 - 4\,(1 - a_0)a_2 + (a_1)^2 = 0$$

$$-12\,(1 - a_0)a_3 + 8a_1a_2 = 0$$

$$\frac{1}{2}\left(36a_1a_3 + 16(a_2)^2\right) = 0$$

$$28a_2a_3 = 0$$

Particularly for n = 3, we have 4 variables. We already have 2 conditions from the boundary values, so we take the first two equations of the above 4 equations obtained.

Problem with this case

So, the 4 equations that we solve are: $a_0 = 1$

$$\sum_{n=1}^{4} a_i = 0$$

$$1 - 4\left(1 - a_0\right)a_2 + a_1^2 = 0$$

$$-12\left(1 - a_0\right)a_3 + 8a_1a_2 = 0$$

When we try to plug in $a_0 = 1$ we arrive at $a_1^2 + 1 = 0$

This Problem is encountered even when using the power series centered around (1,0) and will be encountered while using this power series solution irrespective of the point around which the series is centered

Why is this happening

Consider the equation:

$$1 + (y')^2 - 2y''(1 - y) = 0$$

The solution for $y(x)$ is of the form:

$$y(x) = a_0 + \sum_i a_i x^i$$

Boundary condition $y(0) = 1$, so

$$a_0 = 1$$

This makes our solution:

$$y(x) = 1 + \sum_i a_i x^i$$

Substitute into the original equation:

$$1 + (y')^2 - 2y''(1 - y) = 0$$

$$1 + (y')^2 - 2y'' \left(1 - a_0 + \sum_i a_i x^i\right) = 0$$

Continued

Since $a_0 = 1$, this reduces to:

$$1 + (y')^2 - 2y'' \left(\sum_i a_i x^i \right) = 0$$

Let us call the terms:

$$(1 + (y')^2) \quad \text{Term 1}, \qquad -2y'' \left(\sum_i a_i x^i \right) \quad \text{Term 2}$$

The coefficient-matching equation for x^0 will only get contribution from **Term 1**.

Now, for the coefficients:

$$1 + \text{coeff}_{x^0} \left[(y')^2 \right] = 0$$

Calculate y' and y'':

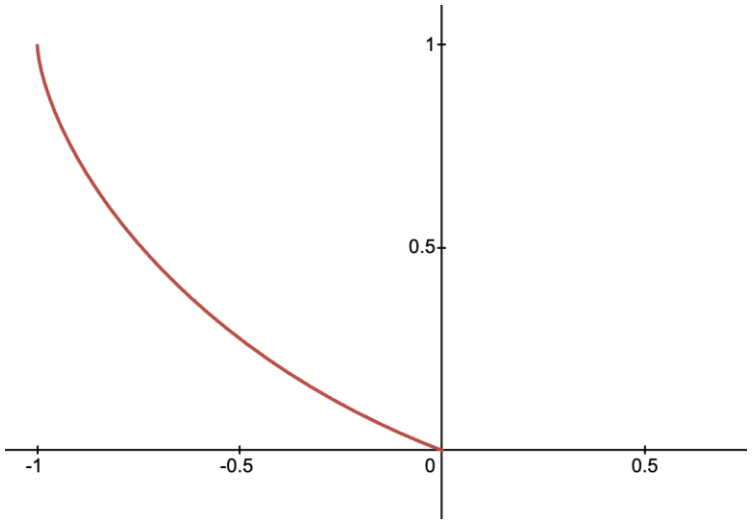
$$y' = \sum_i i a_i x^{i-1}$$

So, $\text{coeff}_{x^0} \left[(y')^2 \right]$ comes from a_1^2.
Thus,

$$1 + a_1^2 = 0 \implies a_1 = \pm i$$

Updated Case: Shifted Brachistochrone

Instead, solve for a shifted Brachistochrone which starts from (-1,1) and passes through (0,0)



So, our conditions now become:

1. $y_B = 1$
2. $y(0) = 0$
3. $y(-1) = 1$

Solution Method

We have 2 equations already

$$y(0) = 0$$

$$y(-1) = 1$$

We take the first n-1 equations of the equations obtained

So, our conditions now become

1. $y_B = 1$
2. $y(0) = 0$
3. $y(-1) = 1$

Example: Solving for second order

The polynomial: $A_2(x) = a_0 + a_1 x + a_2 x^2$

The conditions: $a_0 = 0$ $a_0 - a_1 + a_2 = 1$ $1 - 4(1 - a_0)a_2 + a_1^2 = 0$

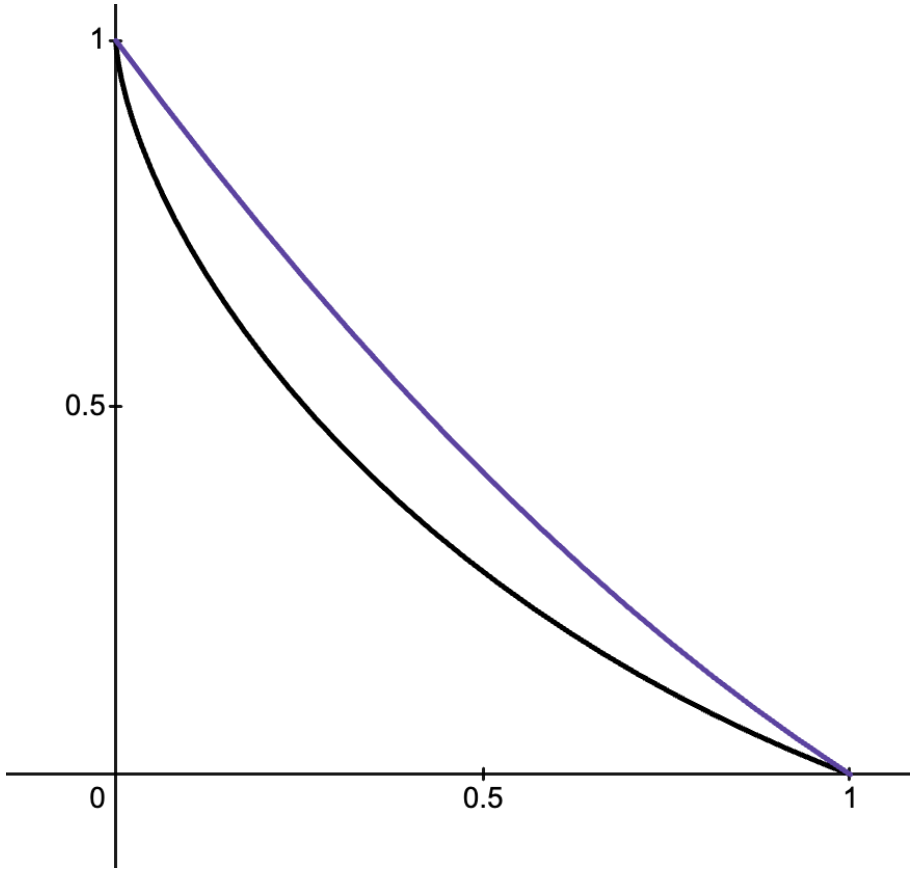
After solving these, we get: $a_1 = -0.64575$ $a_2 = 0.35425$

Since we shifted the Brachistochrone, to shift it back, the required Polynomial should be

$$y = A_2(x - 1)$$

Plot of second order polynomial

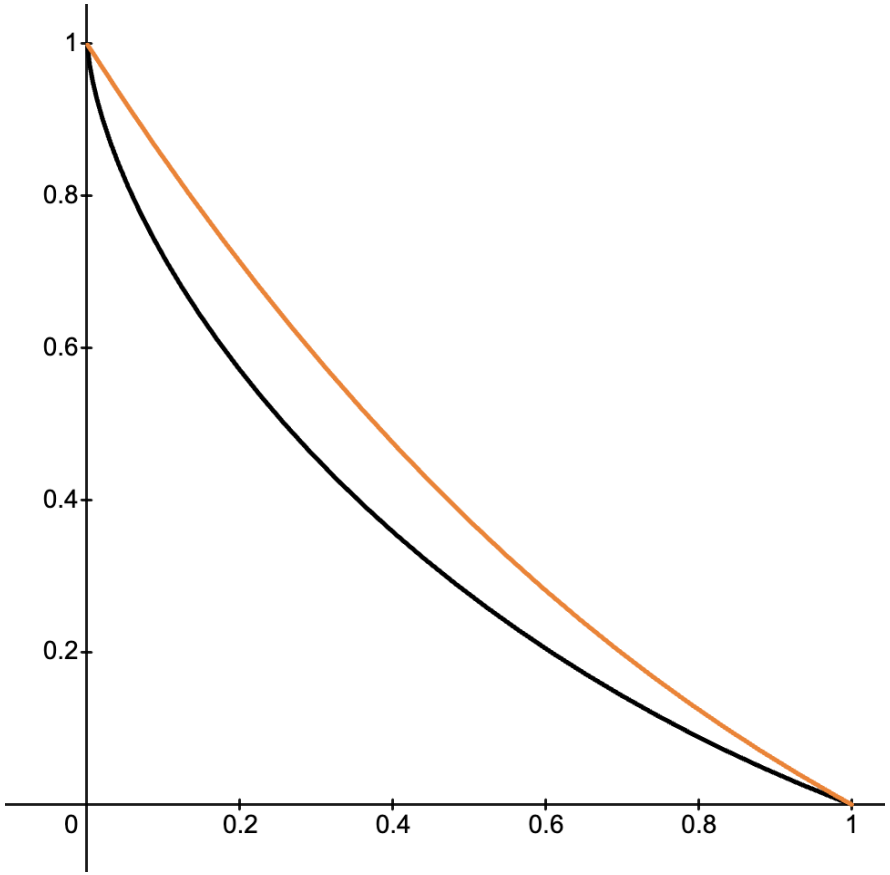
This is the Plot of $y = A_2(x-1)$



Black: Original Brachistochrone
Purple: Second order approximation

Third order polynomial

Similarly, this is the Plot of a third order approximation:



Method/Polynomial in a1

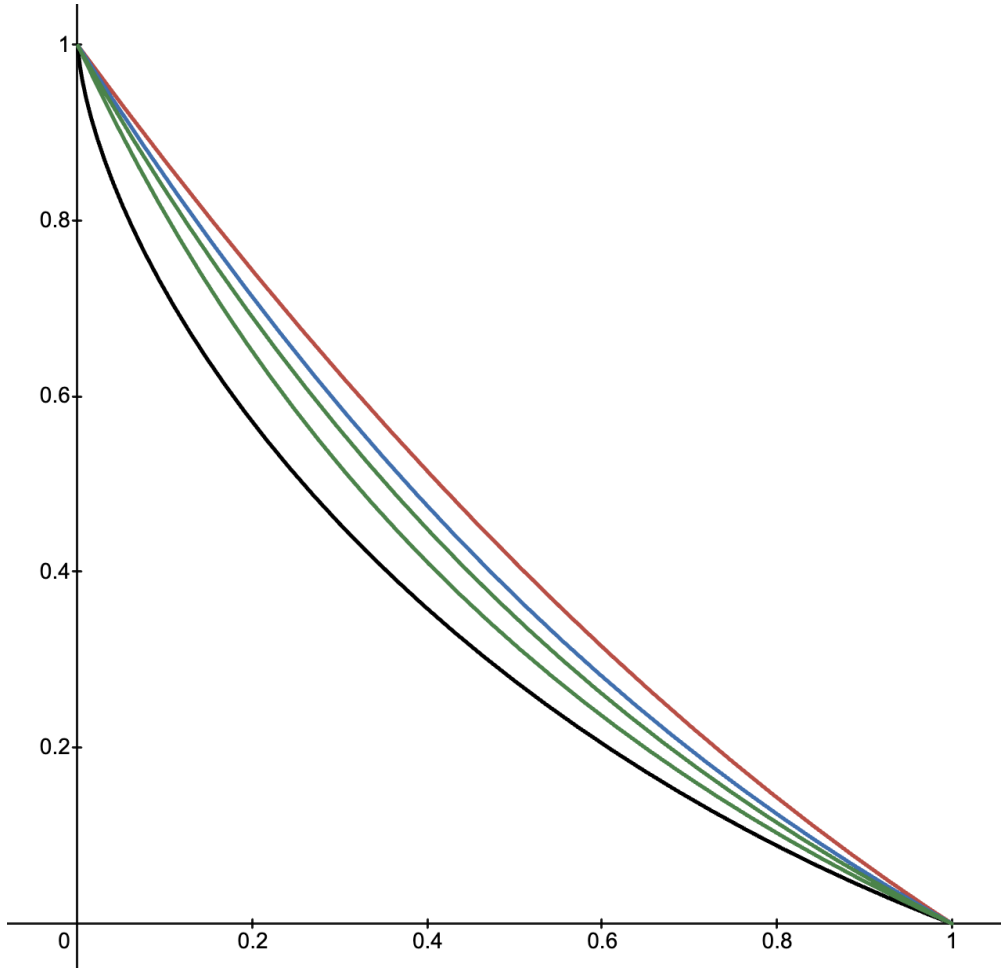
- All the coefficients are represented in terms of a1. $a_2 = \frac{1 + a_1^2}{4} \qquad a_3 = \frac{a_1(1 + a_1^2)}{6}$
- The condition y(-1) = 1 gives the equation:

$$\sum_{i=0}^{n} (-1)^i a_i = 1$$

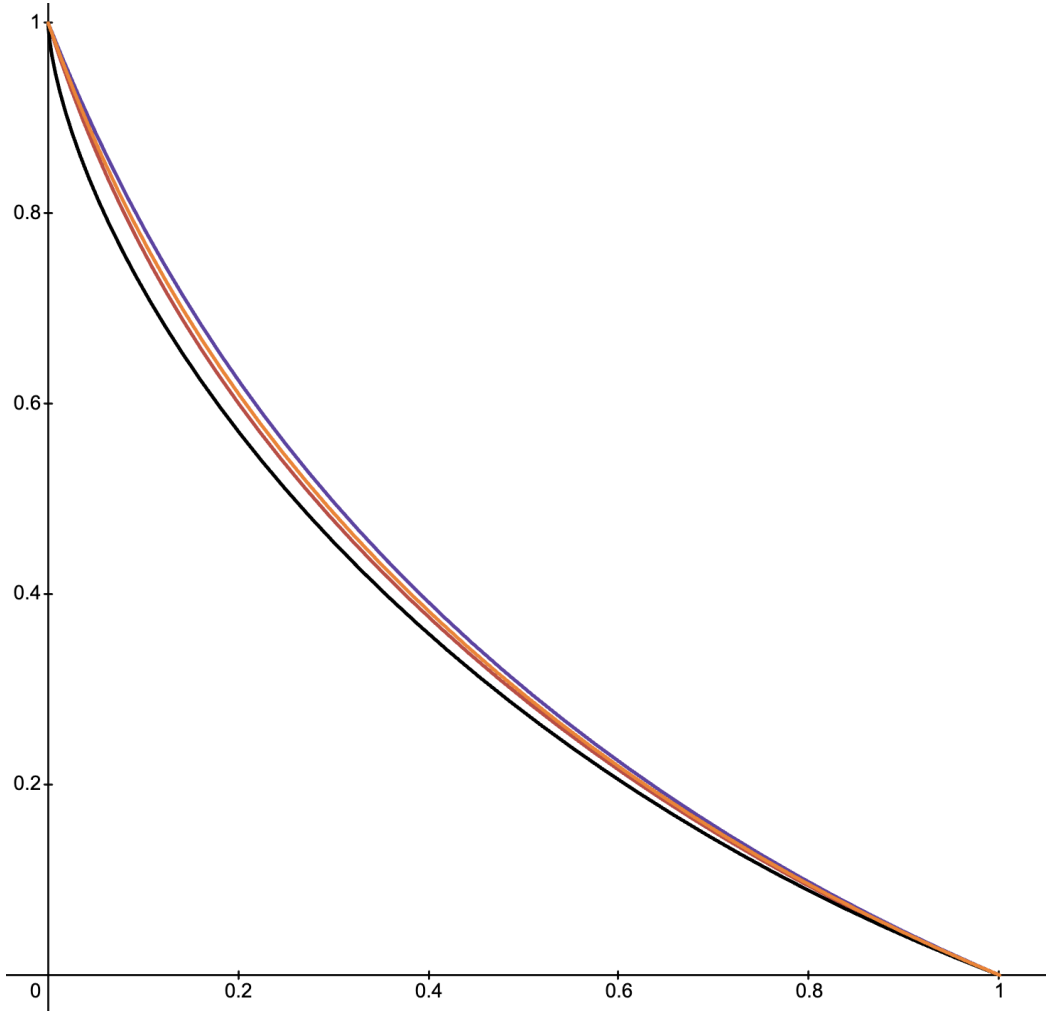
- Putting the ai's in terms of a1, we get an nth order polynomial equation in a1:

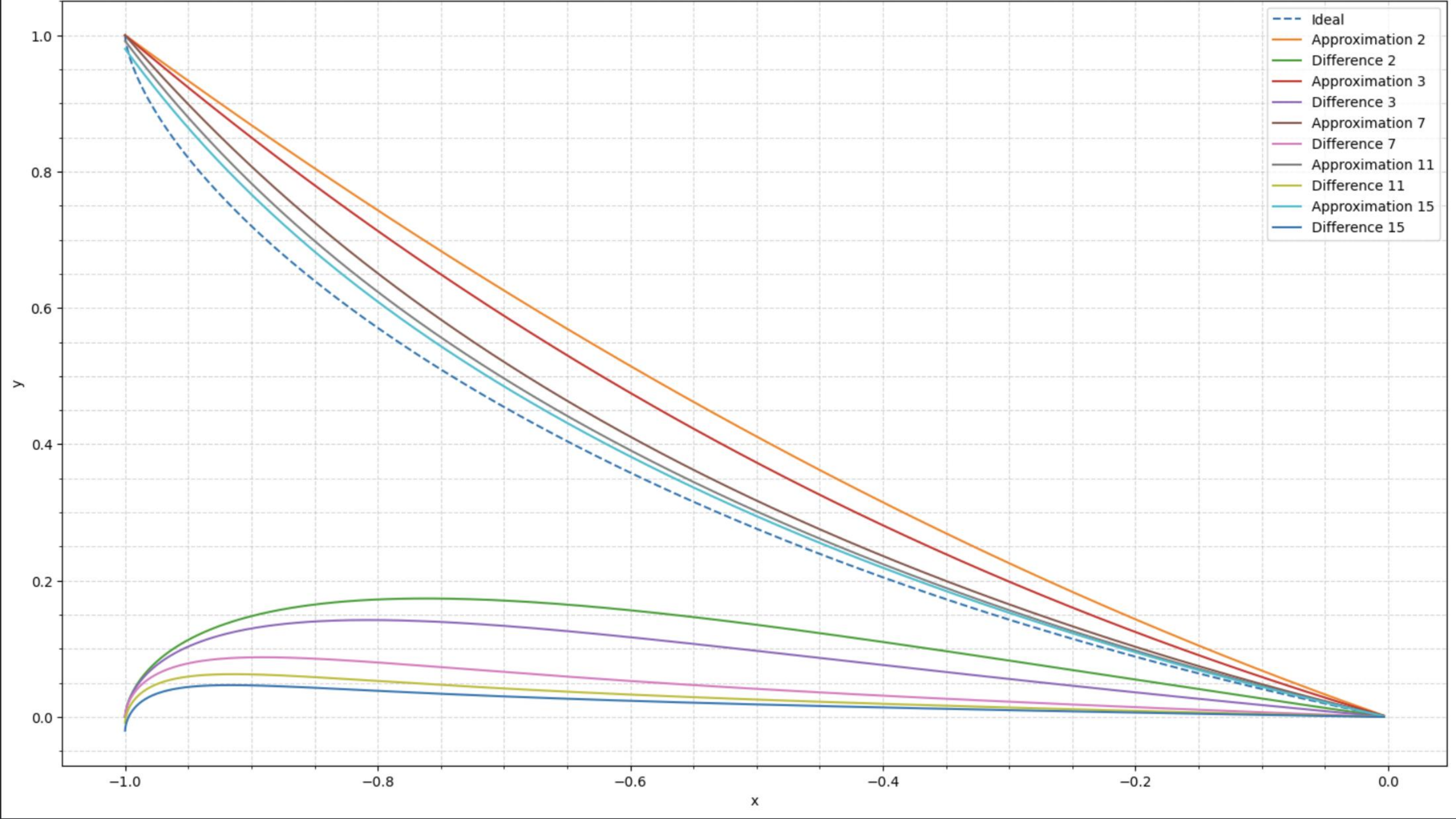
$$c_n a_1^n + c_{n-1} a_1^{n-1} + \cdots + c_1 a_1 + c_0 = 0$$

Plots of 2nd, 3rd, 4th and 7th orders



Plots of 11th, 15th and 20th orders




Ideal
Approximation 2
Difference 2
Approximation 3
Difference 3
Approximation 7
Difference 7
Approximation 11
Difference 11
Approximation 15
Difference 15
1.0
0.8
0.6
0.4
0.2
0.0
y
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
x

Interval Method

- We take the coefficient a1 as an interval $A_1 = [a_1^-, a_1^+]$
- The governing equations we get from coefficient matching are now all in terms of intervals.
- In other words, we get all other coefficients in terms of interval as well, related to the interval A1

$$A_2 = \frac{1 + A_1^2}{4} \qquad A_3 = \frac{A_1(1 + A_1^2)}{6}$$

- The polynomial equation in a1 we found earlier now becomes an interval problem:

$$0 \in c_n A_1^n + c_{n-1} A_1^{n-1} + \cdots + c_1 A_1 + c_0$$

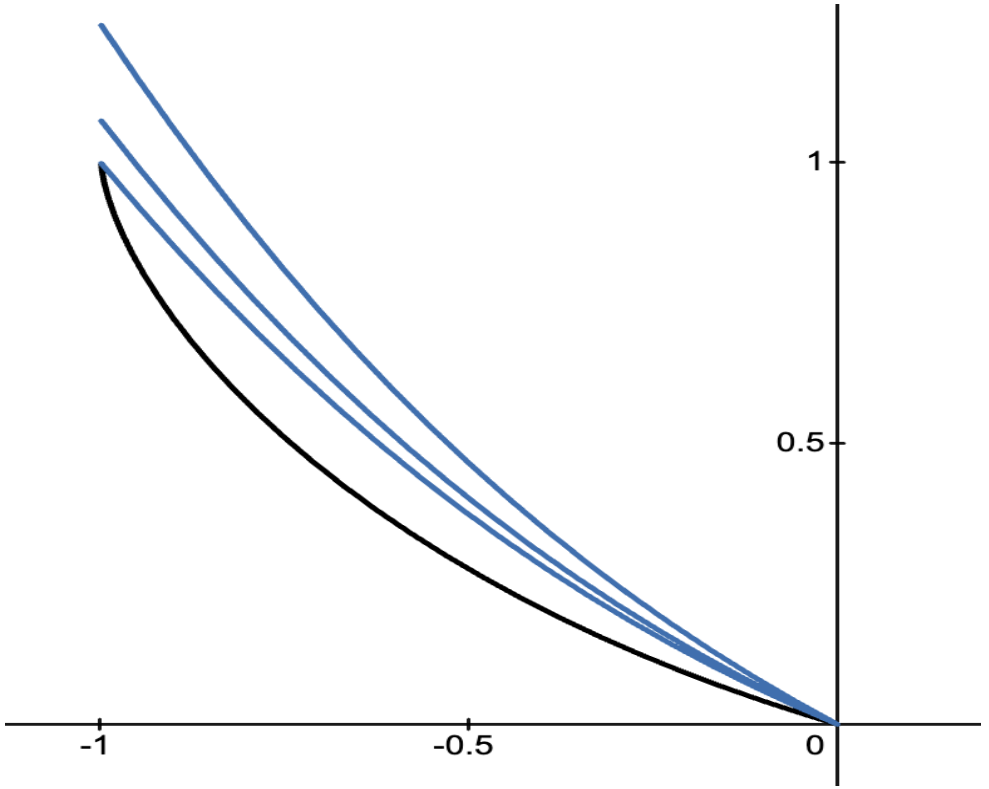
- Solving this involves solving inequalities and obtaining families of upper and lower bounding curves.

The upper bound

- We have the interval problem $0 \in c_n A_1^n + c_{n-1} A_1^{n-1} + \cdots + c_1 A_1 + c_0$
- This arises from the boundary condition y(-1) = 1. However, the upper and lower bound curves might not necessarily pass through (-1, 1).
- Solving it as the non-interval problem $c_n A_1^n + c_{n-1} A_1^{n-1} + \cdots + c_1 A_1 + c_0 = 0$ would give us an upper bound passing through (-1, 1).
- Solving the inequalities would give family of upper bounds which do not pass through (-1, 1).

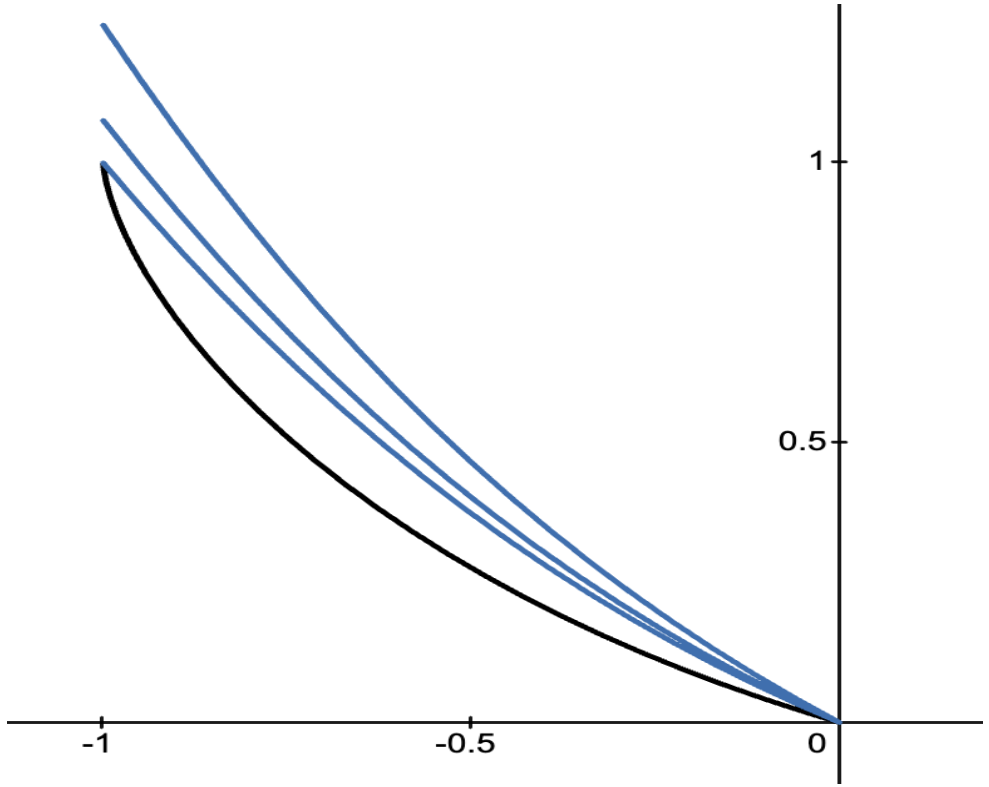
Family of upper bounds

- Particularly, for n = 3, we get the inequality $a_1^- \leq -0.553123$
- Accordingly, these are some of the upper bounding curves:



The tightest upper bound

- Clearly, the tightest upper bound is the one passing through (-1, 1).



- So, for the upper bound we solve the polynomial equation in a1.

The lower bound

- Lower bound never passes through (-1, 1) without intersecting the brachistochrone.
- So, we don’t care about the point solution of the polynomial equation in a1.
- Rather we try to bound y'(0) directly from the differential equation.
- We start with a flexible lower bound interval $y'(0) \in [-\frac{1}{2}, 0]$

$$a_1^+ \in [-\frac{1}{2}, 0]$$

The lower bound – starting interval

- A quick integral and inequalities analysis gives

$$y''(x) = \frac{1 + (y')^2}{2(1-y)} \geq \frac{1}{2} \qquad (-1 < x \leq 0).$$

$$a_1^+ \in [-\frac{1}{2}, 0]$$

$y'' \geq 0$ on $(-1, 0]$, so y' is strictly increasing on $(-1, 0]$.

$$\int_{-1}^{0} y'(x)\,dx = y(0) - y(-1) = 0 - 1 = -1.$$

$$y'(-1) \leq \frac{1}{0-(-1)} \int_{-1}^{0} y'(x)\,dx \leq y'(0),$$

so $y'(-1) \leq -1$ and $y'(0) \geq -1$.

$$y'(0) - y'(-1) = \int_{-1}^{0} y''(x)\,dx \geq \int_{-1}^{0} \frac{1}{2}\,dx = \frac{1}{2}$$

$$y'(0) \geq y'(-1) + \frac{1}{2} \geq -1 + \frac{1}{2} = -\frac{1}{2}.$$

Improving the lower bound

- Doing some mathematical manipulations, we get an integral equation as a function of a_1^+

$$\text{Let } p(x) = y'(x) \text{ and } p_0 = y'(0)$$

$$p' = \frac{1+p^2}{2(1-y)}.$$

$$p\frac{dp}{dy} = \frac{1+p^2}{2(1-y)}$$

$$\int_{p_0}^{p(y)} \frac{2s}{1+s^2}\,ds = \int_0^y \frac{dt}{1-t}.$$

$$\ln\frac{1+p(y)^2}{1+p_0^2} = -\ln(1-y)1+p(y)^2 = \frac{1+p_0^2}{1-y}$$

$$p(y) = -\sqrt{\frac{p_0^2+y}{1-y}}$$

Improving the lower bound

$$\frac{dx}{dy} = \frac{1}{p(y)} = -\sqrt{\frac{1-y}{p_0^2+y}}$$

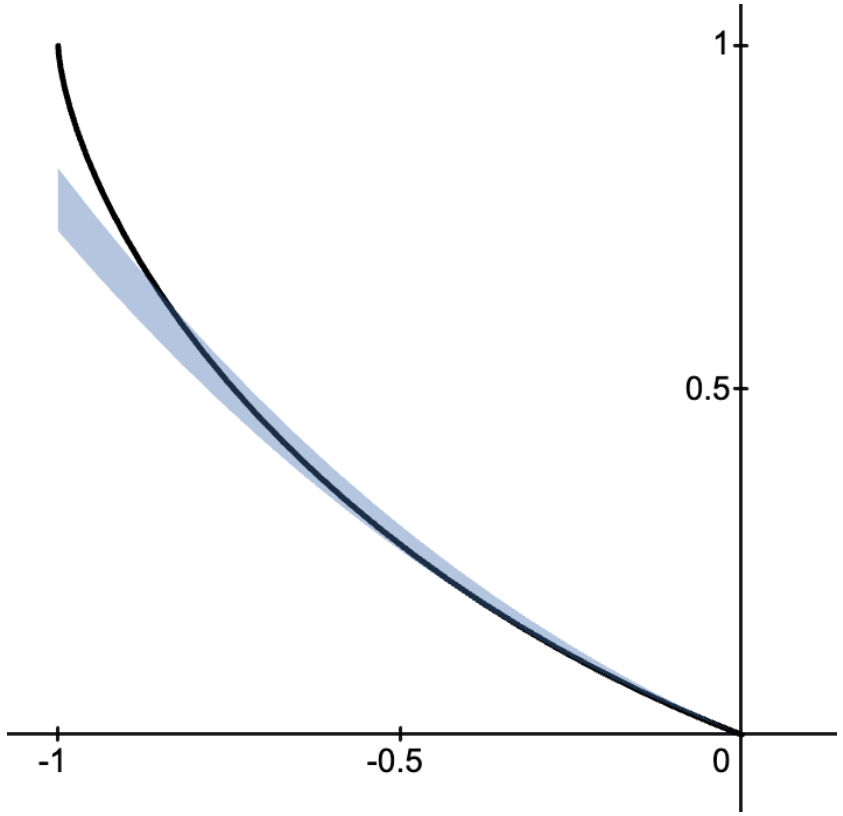
$$x(1) - x(0) = \int_0^1 \frac{dx}{dy}\, dy = -\int_0^1 \sqrt{\frac{1-y}{p_0^2+y}}\, dy$$

$$I(p_0) := \int_0^1 \sqrt{\frac{1-y}{p_0^2+y}}\, dy = 1$$

- Starting with the flexible interval, we iteratively solve this equation.
- With each iteration, we get a better bound of $p_0 = y'(0) = a_1^+$
- As a result, we get a nicer and tighter lower bound.

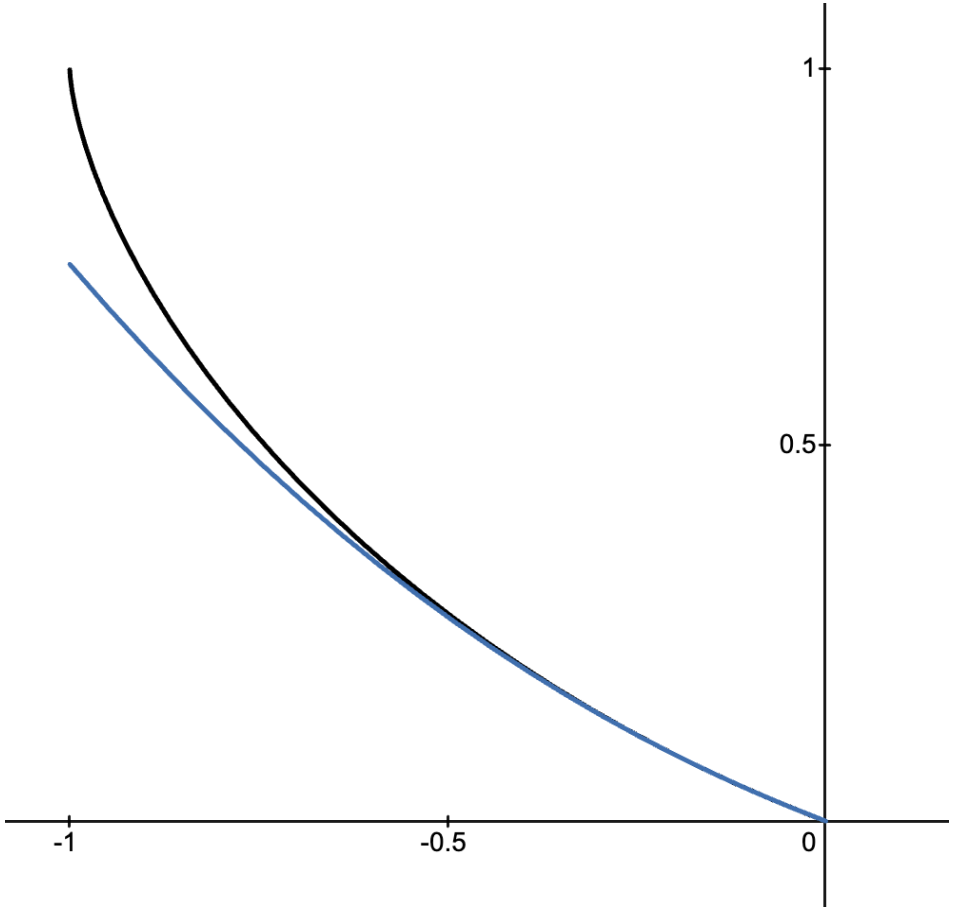
Improved lower bound - example

- For n = 3, after 3 iterations we get $a_1^+ \in [-0.4375,\ -0.375]$



Improved lower bound - example

- For n = 3, after 14 iterations we get $a_1^+ \in [-0.3818969,\ -0.3818664]$



Algorithm for solving for the bounds of a1 from Integral equation using QuadGK.jl package

QuadGK package implements one-dimensional numerical integration ("quadrature") in Julia using adaptive Gauss–Kronrod quadrature. For the implementation we will be using the function 'quadgk' This function gives us error bounds of the integral in addition to point-wise solution

For our Algorithm performing Binary search along our already known bounds for a1 [-0.5,0]

```
function I_of_p(p)
    integrand(y) = sqrt((1 - y) / (p^2 + y))
    val, err = quadgk(integrand, 0.0, 1.0; rtol=1e-6, atol=1e-6)
    return val, err
end
```

```
function solve_p0(a, b; tol=1e-10, maxiter=50)
    fa = I_of_p(a) - 1
    fb = I_of_p(b) - 1
    if fa * fb > 0
        error
    end
    for iter in 1:maxiter
        m = (a + b) / 2
        fm = I_of_p(m) - 1
        @info "iter=$iter  p=$m   I(p)-1=$(fm)"
        if abs(fm) < tol
            return m
        end
        if fa * fm < 0
            b = m
            fb = fm
        else
            a = m
            fa = fm
        end
    end
    return (a + b) / 2
end
```

Results of the Binary Search Iterations (1)

```
Iter 1: m = -0.2500000000   Integral[m] ∈ [1.158680663755, 1.158681922544]
Iter 2: m = -0.3750000000   Integral[m] ∈ [1.007466235589, 1.007467342584]
Iter 3: m = -0.4375000000   Integral[m] ∈ [0.942607776390, 0.942608638847]
Iter 4: m = -0.4062500000   Integral[m] ∈ [0.974220256956, 0.974221184865]
Iter 5: m = -0.3906250000   Integral[m] ∈ [0.990633412092, 0.990634406428]
Iter 6: m = -0.3828125000   Integral[m] ∈ [0.998996646396, 0.998997689535]
Iter 7: m = -0.3789062500   Integral[m] ∈ [1.003218055968, 1.003219128877]
Iter 8: m = -0.3808593750   Integral[m] ∈ [1.001104016294, 1.001105073816]
Iter 9: m = -0.3818359375   Integral[m] ∈ [1.000049499041, 1.000050549251]
Iter 10: m = -0.3823242188   Integral[m] ∈ [0.999522864820, 0.999523911464]
Iter 11: m = -0.3820800781   Integral[m] ∈ [0.999786129933, 0.999787178353]
Iter 12: m = -0.3819580078   Integral[m] ∈ [0.999917801485, 0.999918850798]
Iter 13: m = -0.3818969727   Integral[m] ∈ [0.999983647012, 0.999984696773]
Iter 14: m = -0.3818664551   Integral[m] ∈ [1.000016572214, 1.000017622199]
Iter 15: m = -0.3818817139   Integral[m] ∈ [1.000000109410, 1.000001159283]
Iter 16: m = -0.3818893433   Integral[m] ∈ [0.999991878160, 0.999992927977]
Iter 17: m = -0.3818855286   Integral[m] ∈ [0.999995993772, 0.999997043617]
Iter 18: m = -0.3818836212   Integral[m] ∈ [0.999998051588, 0.999999101447]
Iter 19: m = -0.3818826675   Integral[m] ∈ [0.999999080498, 1.000000130364]
Iter 20: m = -0.3818821907   Integral[m] ∈ [0.999999594954, 1.000000644823]
Iter 21: m = -0.3818824291   Integral[m] ∈ [0.999999337726, 1.000000387593]
Iter 22: m = -0.3818823099   Integral[m] ∈ [0.999999466340, 1.000000516208]
Iter 23: m = -0.3818822503   Integral[m] ∈ [0.999999530647, 1.000000580516]
Iter 24: m = -0.3818822801   Integral[m] ∈ [0.999999498493, 1.000000548362]
Iter 25: m = -0.3818822950   Integral[m] ∈ [0.999999482417, 1.000000532285]
Iter 26: m = -0.3818823025   Integral[m] ∈ [0.999999474378, 1.000000524247]
Iter 27: m = -0.3818822987   Integral[m] ∈ [0.999999478398, 1.000000528266]
Iter 28: m = -0.3818823006   Integral[m] ∈ [0.999999476388, 1.000000526256]
Iter 29: m = -0.3818823015   Integral[m] ∈ [0.999999475383, 1.000000525251]
Iter 30: m = -0.3818823020   Integral[m] ∈ [0.999999474881, 1.000000524749]
Iter 31: m = -0.3818823018   Integral[m] ∈ [0.999999475132, 1.000000525000]
Converged at iter 31: root ≈ -0.3818823017645627; Integral ∈ [0.999999475131918
8, 1.0000005250002422]

Root lies in [-0.38188230199739337, -0.3818823015317321] with last Integral ∈ [
0.9999994751319188, 1.0000005250002422]
```

Results of the Binary Search Iterations (2)

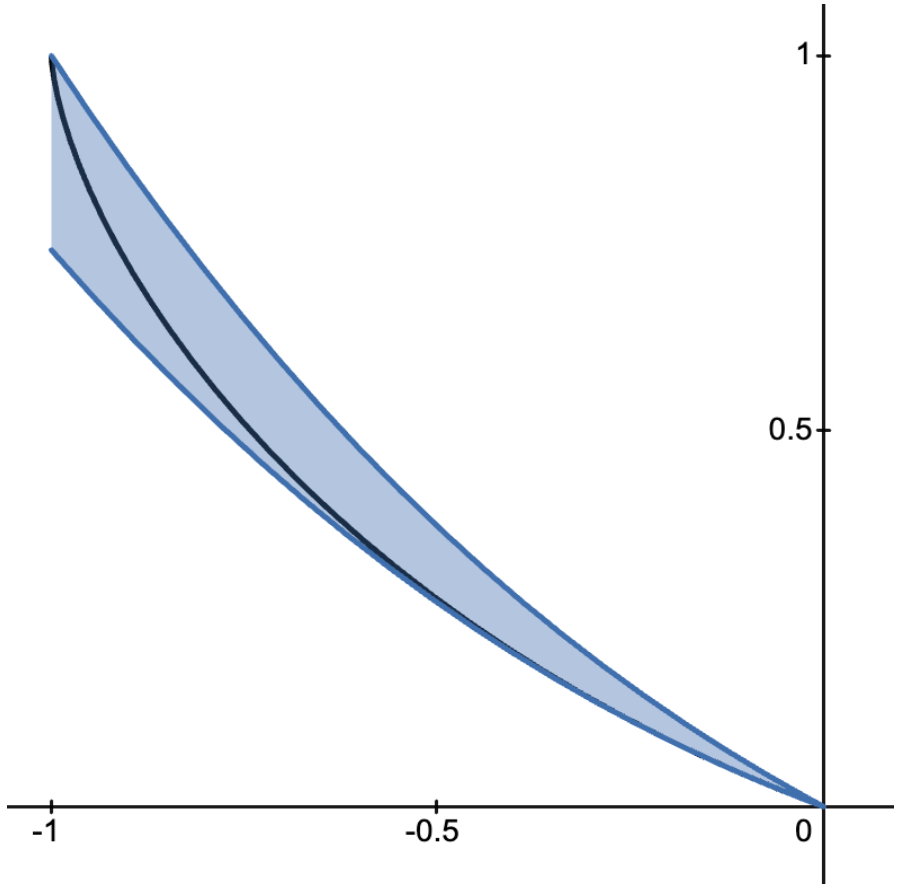
```
iter=1   p=-0.25    I(p)-1=0.15868126764731905
iter=2   p=-0.375   I(p)-1=0.007466764473069043
iter=3   p=-0.4375   I(p)-1=-0.05739181646387226
iter=4   p=-0.40625   I(p)-1=-0.025779303442926116
iter=5   p=-0.390625   I(p)-1=-0.009366115225125426
iter=6   p=-0.3828125   I(p)-1=-0.0010028565840267678
iter=7   p=-0.37890625   I(p)-1=0.0032185678411871255
iter=8   p=-0.380859375   I(p)-1=0.0011045204898438854
iter=9   p=-0.3818359375   I(p)-1=4.999958838403096e-5
iter=10   p=-0.38232421875   I(p)-1=-0.0004766364116343125
iter=11   p=-0.382080078125   I(p)-1=-0.00021337041223412623
iter=12   p=-0.3819580078125   I(p)-1=-8.169841484750862e-5
iter=13   p=-0.38189697265625   I(p)-1=-1.5852664308591002e-5
iter=14   p=-0.381866455078125   I(p)-1=1.7072649225013947e-5
iter=15   p=-0.3818817138671875   I(p)-1=6.097892604195465e-7
iter=16   p=-0.38188934326171875   I(p)-1=-7.6214883230063535e-6
iter=17   p=-0.3818855285644531   I(p)-1=-3.505862230746004e-6
iter=18   p=-0.3818836212158203   I(p)-1=-1.4480396601790346e-6
iter=19   p=-0.3818826675415039   I(p)-1=-4.1912599380022897e-7
iter=20   p=-0.3818821907043457   I(p)-1=9.533143474627082e-8
iter=21   p=-0.3818824291229248   I(p)-1=-1.6189732876537022e-7
iter=22   p=-0.38188230991363525   I(p)-1=-3.328295916649182e-8
iter=23   p=-0.3818822503089905   I(p)-1=3.1024234514731575e-8
iter=24   p=-0.38188228011131287   I(p)-1=-1.1293628254804844e-9
iter=25   p=-0.3818822652101517   I(p)-1=1.4947435511558638e-8
iter=26   p=-0.38188227266073227   I(p)-1=6.909036454061379e-9
iter=27   p=-0.38188227638602257   I(p)-1=2.8898365922458424e-9
iter=28   p=-0.3818822782486677   I(p)-1=8.802367723603766e-10
iter=29   p=-0.3818822791799903   I(p)-1=-1.2456324860465884e-10
iter=30   p=-0.381882278714329   I(p)-1=3.778368729001613e-10
iter=31   p=-0.38188227894715965   I(p)-1=1.2663670112544878e-10
iter=32   p=-0.38188227906357497   I(p)-1=1.0367262603949712e-12

Solution p(0) = -0.38188227906357497
```

I(p) is pointwise value of Integral at a particulat iteration

Upper and lower bounds for n = 3

- Plot of both upper and lower bound for n = 3 after sufficient number of iterations:



Thank You

Comparison of QuadGK.jl vs Matlab Quadrature function

Matlab Inbuild function

```
>> f = @(x) sin(x + exp(x));
>> [q, errbnd] = quad(f, 0, 8);
>> fprintf('Integral ≈ %.10f\n', q);
Integral ≈ 0.2511027220
```

QuadGk function for calulating rigorous bounds of the Integral

```
begin
    using QuadGK
    using Printf

    f(x) = sin(x + exp(x))
    val, err = quadgk(f, 0.0, 8.0)
    @printf("Integral : %.10f\n", val)
    @printf("Error bound: %.10e\n", err)
    @printf("Interval containing the integral: [%.10f, %.10f]\n", val - err, val +
    err)
end
```

```
Integral : 0.3474001727
Error bound: 5.0976381923e-09
Interval containing the integral: [0.3474001676, 0.3474001778]
```