Optimal Control Using Interval Analysis

Under supervision of

Prof. Shaunak Sen

Vangapally Santhoshi

M.Tech Control and Automation

Motivation

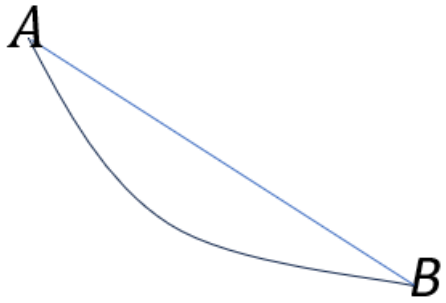
- The motivation behind using interval analysis approach to optimal control problems which helps in finding all feasible solutions of the optimal control problem.

Problem Statement

- In this the primary objective is to apply interval analysis to brachistochrone problem for bounding the minimum time trajectories.

Brachistochrone curve

- The Brachistochrone problem is a classic problem in calculus of variations
- It seeks to determine the shape of a curve along which a particle will slide without friction from one point to another in the shortest time, under the influence of gravity.



Mathematical Formulation

- The law of conservation of energy can be applied to analyze and calculate the time in problems where energy is conserved, such as a body moving under the influence of gravity the total energy is the sum of kinetic energy (KE) and potential energy (PE) remains constant.

$$mgh = \frac{1}{2} * m * {v_i}^2 + m * g * y_i$$

$$v_i = \sqrt{2 * g * (h - y_i)} \qquad \mathrm{t} = l / v_i$$

Analytical solution

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

$$t = \int \sqrt{\frac{dx^2 + dy^2}{2 * g * (h - y)}}$$

The analytical solution which is known to be a cycloid curve

$$x = c_1(2\theta + \sin(2\theta)) + c_2$$

$$y = 1 - c_1(1 + \cos(2\theta))$$

Here, θ is a parameter varying from θ_0 to θ_f , and c_1, c_2 are constants chosen such that the curve passes through the given boundary points.

- Boundary conditions : $(x_0, y_0) = (0,1)$ and $(x_f, y_f) = (1,0)$

$$\theta_0 = \frac{\pi}{2}$$

$$c_2 = -c_1 * \pi$$

$$c_1 = \frac{1}{1+\cos(2*\theta_f)}$$

$$f(\theta_f) = 1+\cos(2\theta_f) - (2\theta_f - \pi + sin(2\theta_f)) = 0$$

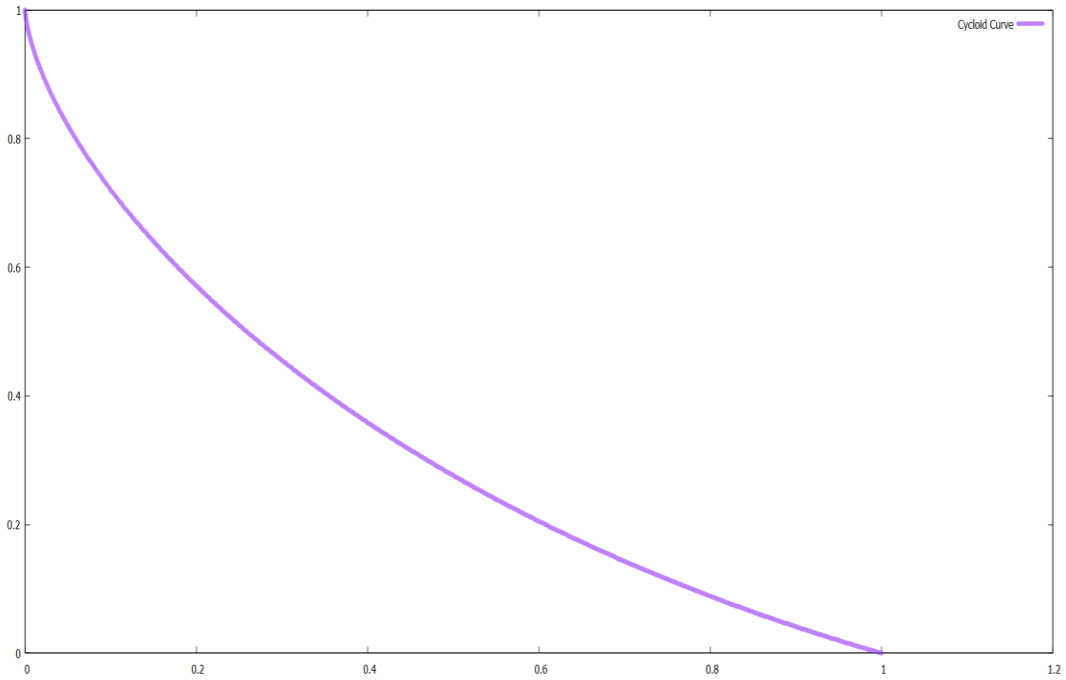
The above equation is solved using interval newton method, with initial guess as $\left[\frac{\pi}{2}, 2\pi\right]$ the solution obtained as θ_f = $[2.7768, 2.77681]$

$$mid(\theta_f) = 2.77680189$$

$$t = \int_{\theta_f}^{\theta_0} \sqrt{\frac{dx^2+dy^2}{2*g*(h-y)}}$$

The minimum time is given by $= \int_{\theta_f}^{\theta_0} \sqrt{\frac{4{c_1}^2(1+\cos 2\theta)^2+4{c_1}^2 \sin^2 2\theta}{2*g*(1+\cos 2\theta)}}\, d\theta$

t = 0.582334sec



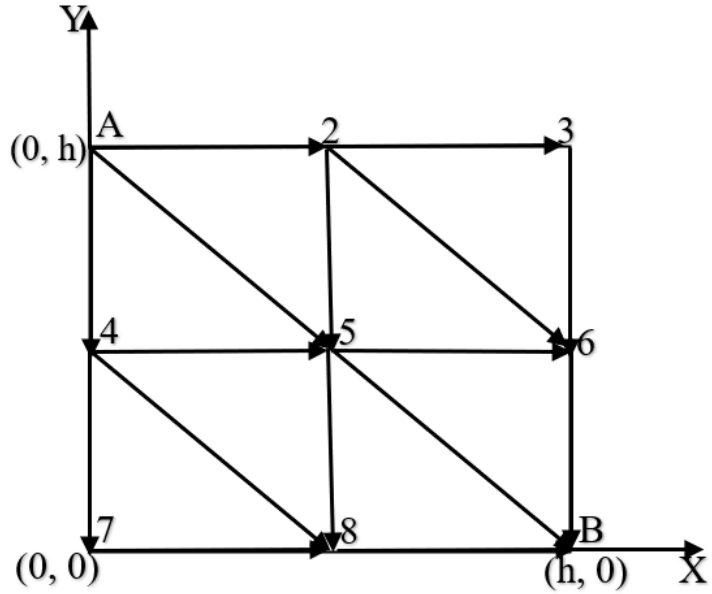
Dynamic programming

- The principle of optimality is a fundamental aspect of dynamic programming, which states that the optimal solution to a dynamic optimization problem can be found by combining the optimal solutions to its sub-problems.

Routing

- To apply the dynamic programming approach to the brachistochrone problem, the continuous state space must first be discretized.

$$t_{AB}^{*} = \min(t_{A5} + t_{5B}^{*}, t_{A4} + t_{4B}^{*}, t_{A2} + t_{2B}^{*})$$
$$t_{4B}^{*} = \min(t_{47} + t_{78} + t_{8B}, t_{45} + t_{5B}^{*}, t_{48} + t_{8B})$$
$$t_{5B}^{*} = \min(t_{5B}, t_{36} + t_{58} + t_{8B}, t_{56} + t_{6B})$$



Interval Dynamic Programming

- Each state is interval, and $t = \left[\frac{d}{v_h}, \frac{d}{v_l}\right]$
- $t_{5B}^{*} = \min(t_{5B}, t_{36} + t_{58} + t_{8B}, t_{56} + t_{6B})$

  $= \min\left(\left[\frac{d_{5B}}{v_{5Bh}}, \frac{d_{5B}}{v_{5Bl}}\right], \left[\frac{d_{58}}{v_{58h}} + \frac{d_{8B}}{v_{8Bh}}, \frac{d_{58}}{v_{58l}} + \frac{d_{8B}}{v_{8Bl}}\right], \left[\frac{d_{56}}{v_{56h}} + \frac{d_{6B}}{v_{6Bh}}, \frac{d_{56}}{v_{56l}} + \frac{d_{6B}}{v_{6Bl}}\right]\right)$

Pluto.jl

aphs\brachistochrone_DP_interval.jl

```
using Plots , PlutoUI
```

```
PlotlyBackend()
```

```
plotly()
```

52.7 μs

```
0.29990812607480277
```

```
begin
    local TimeTaken = NaN*ones(N+1, N+1);
    local Direction = NaN*ones(N+1, N+1);

    for i = 0:N # Rows

        for j = 0:N # Columns

            if (N - j == N) && (N - i == N) # End

                TimeTaken[N - i + 1, N - j + 1] = 0;
                Direction[N - i + 1, N - j + 1] = 0;

            elseif !(N - j == N) && (N - i == N) # Bottom most

                TimeTaken[N - i + 1, N - j + 1] = TimeTaken[N - i + 1, N - j + 2] +
                1/√(N - i);

                Direction[N - i + 1, N - j + 1] = 1; # Go Right

            elseif (N - j == N) && !(N - i == N) # Rightmost

                TimeTaken[N - i + 1, N - j + 1] = TimeTaken[N - i + 2, N - j + 1] +
                2/(√(N - i) + √(N - i + 1));

                Direction[N - i + 1, N - j + 1] = 2; # Go Down
```

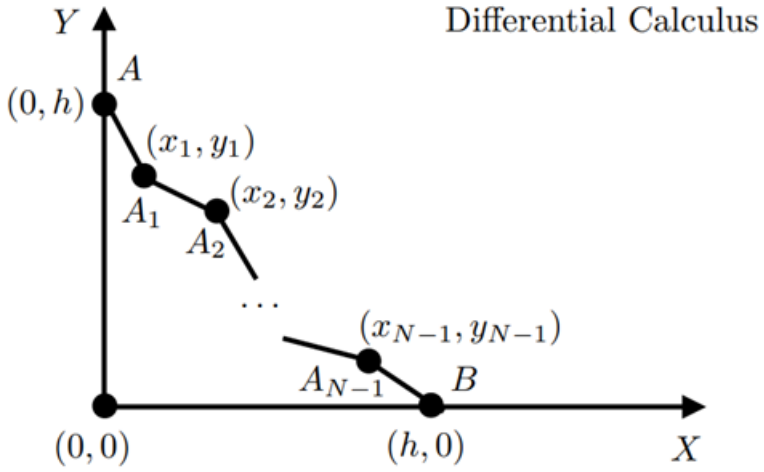
Live Docs Status

Curve Discretization

- Discretization involves breaking down the continuous trajectory into a finite number of points or segments, the minimization problem is a problem in differential calculus, where the partial derivatives with respect to the y_i is equal to

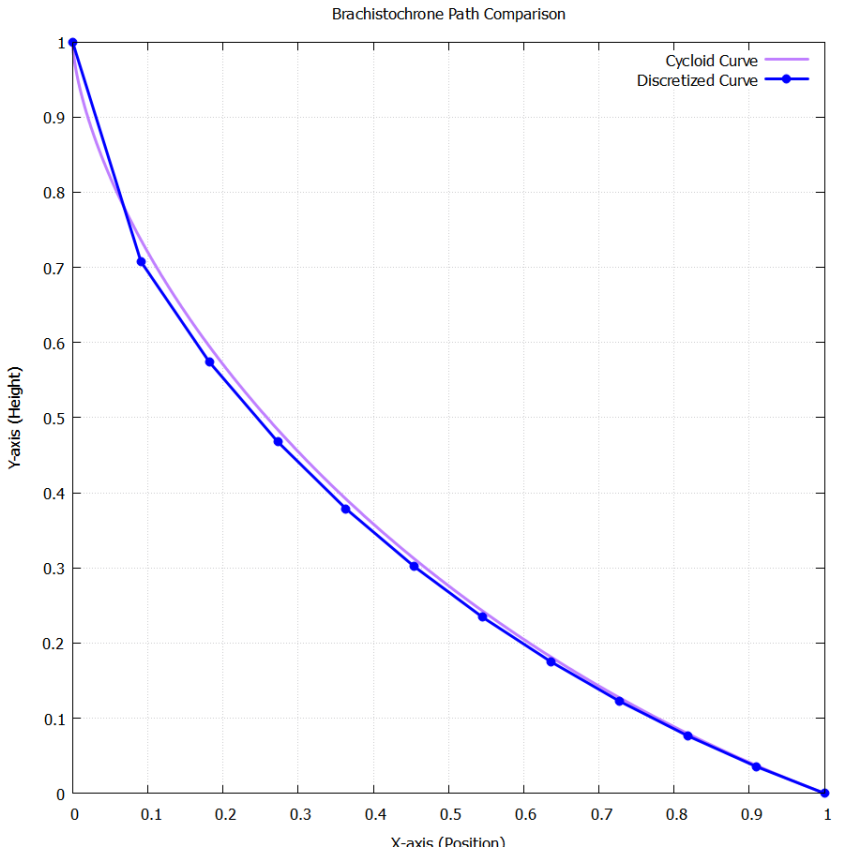
$$t = \frac{\sqrt{(x_1)^2+(1-y_1)^2}}{0.5*(\sqrt{2g(1-y_1)})} + \frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{0.5*(\sqrt{2g(1-y_1)}+\sqrt{2g(1-y_2)})}+\ldots$$

$$\frac{\partial t}{\partial y_1} = 0, \frac{\partial t}{\partial y_2} = 0 \ldots$$



Newton method

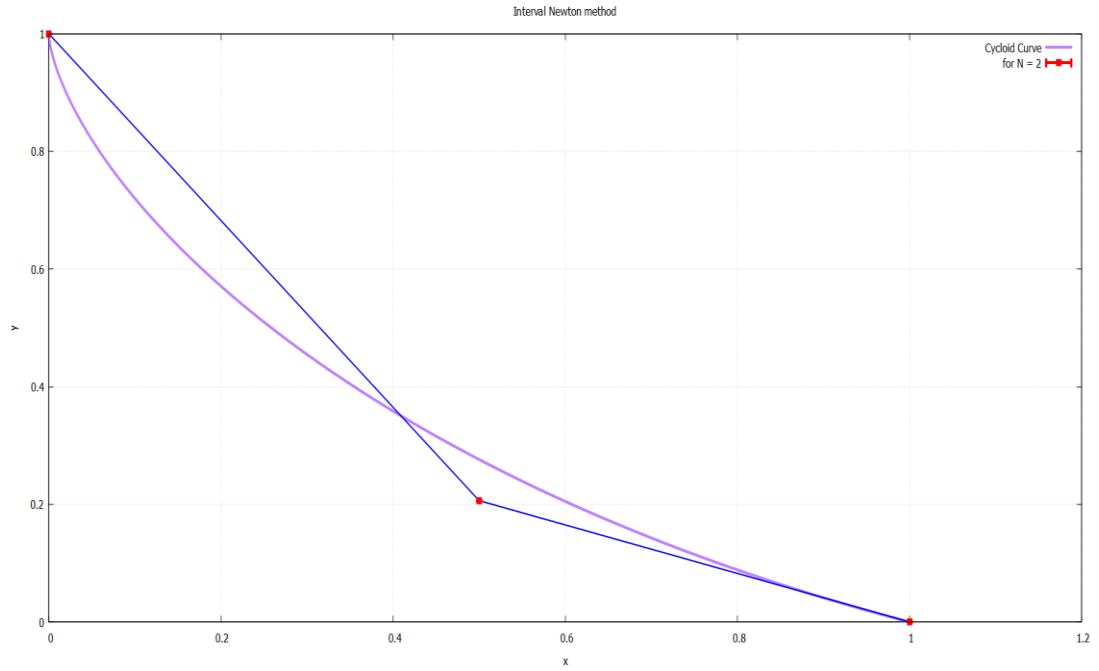
1. **Define the cost function** F(y).
2. **Initialize guess** y0.
3. Find fdash which is the first order derivative of cost function.
4. **Compute the Jacobian matrix** J.
5. **Newton's update step** : ynew = y0 – inv(J)*F(y0)
6. **Check the stopping condition**:
   - If ||Fdash(y0)|| < 0.0001 stop and return ynew.
   - Otherwise, set y0 = ynew and repeat from step 4



Interval Newton method

Interval Newton's update step : N(y) = mid(y0) – inv(J)*Fdash(mid(y0))

ynew = y0 Ω N(y)



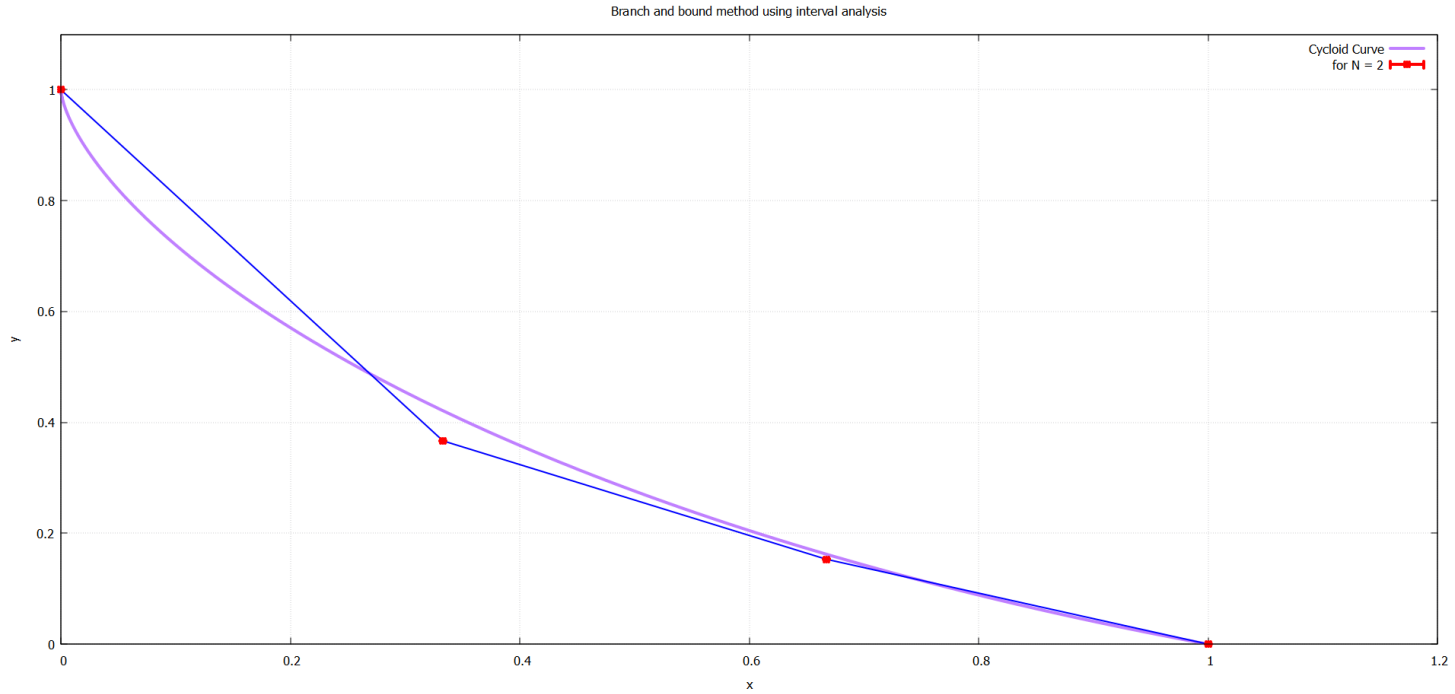
Branch and Bound Algorithm

- The method systematically divides the problem's domain into smaller subdomains (branching) and computes bounds on the objective function within these subdomains (bounding). Subdomains that cannot contain the optimal solution are discarded, thereby narrowing the search space.

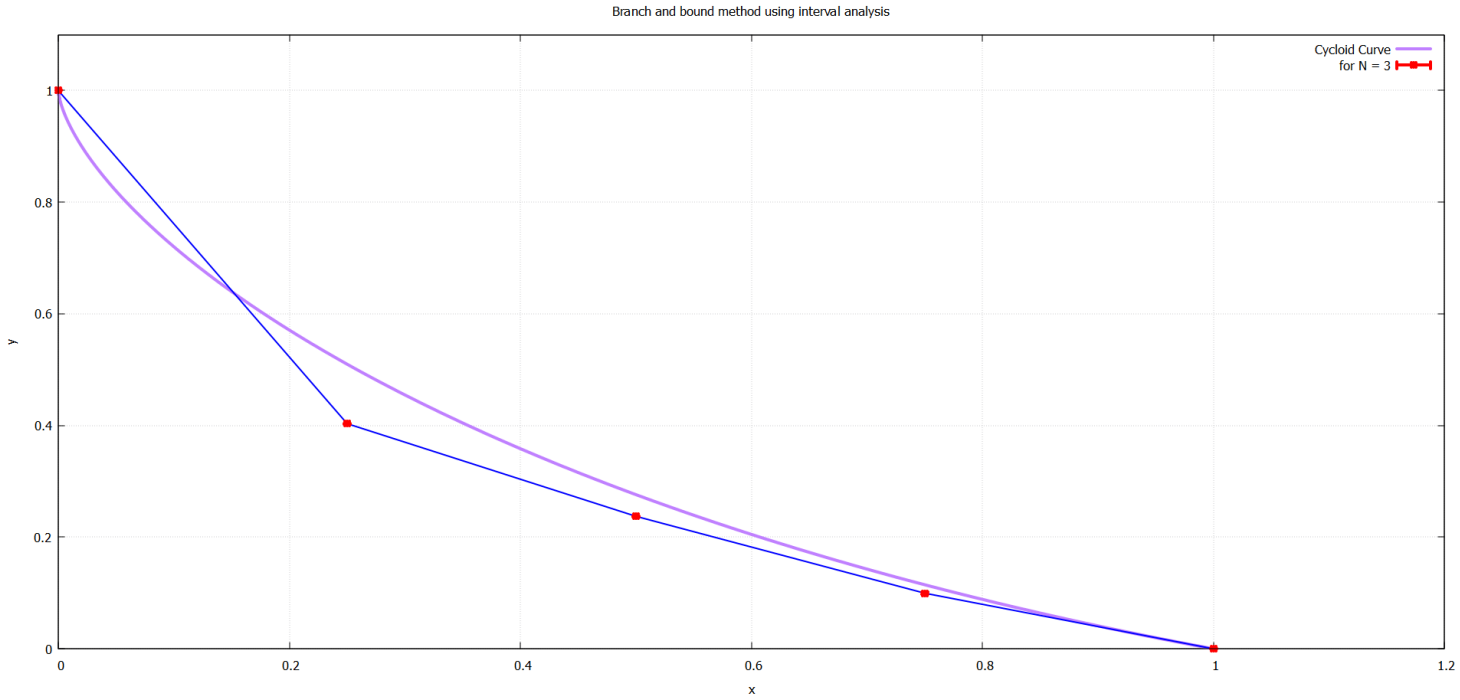
  upperbound = min(upperbound,sup(mid(f(x)))

  if inf(f(x)) <= upperbound ,bisect else discard

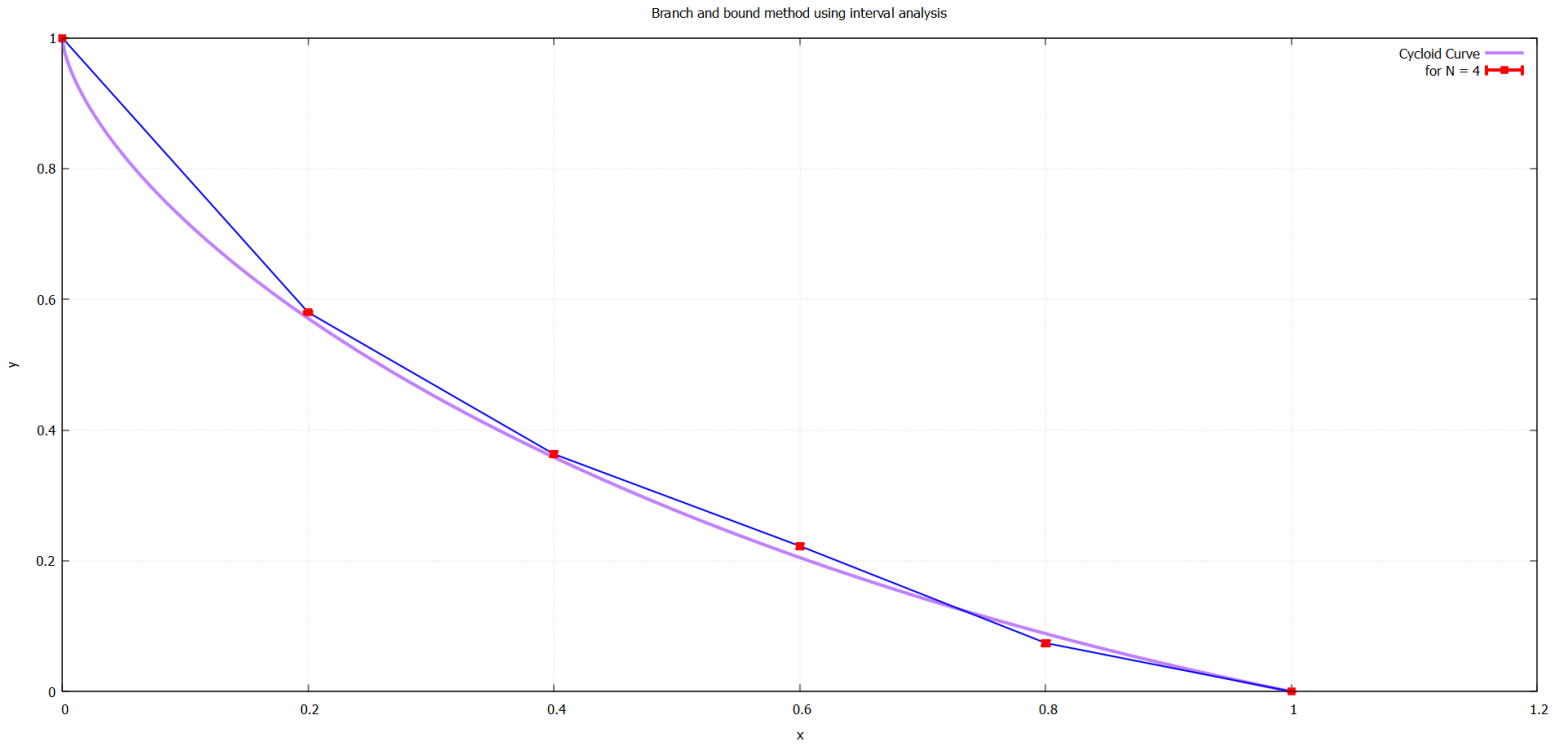
- For n=2, with x1, x2 = 0.333, 0.667 the corresponding value of y1 = [0.36621, 0.367188],y2 = [0.152343, 0.154297] and the minimum time taken = [0.596001, 0.597564]



- For N=3, with x1, x2, x3 = 0.25, 0.5, 0.75 the corresponding value of y1= [0.402343, 0.404297], y2 = [0.236328, 0.238282], y3 = [0.0976562, 0.101563] and the minimum time taken = [0.592065, 0.595677]



- With x1, x2, x3, x4 = 0.2, 0.4, 0.6, 0.8 the corresponding valueof y1 = [0.578124, 0.582032], y2 = [0.359374, 0.367188], y3 = [0.218749, 0.226563], y4 =[0.0703124, 0.0781251] and the minimum time taken = [0.585756, 0.598993]



Conclusion

- In this work, a study of the brachistochrone problem was carried out using methods based on interval analysis. By using interval-based dynamic programming, interval Newton method and branch-bound algorithm

Thank you