The brachistochrone Problem:

Monday, December 21, 2020 6:34 PM

The brachistochrone problem

The brachistochrone problem was posed by Johann Bernoulli in *Acta Eruditorum* Ⓣ in June 1696. He introduced the problem as follows:-

> *I, Johann Bernoulli, address the most brilliant mathematicians in the world. Nothing is more attractive to intelligent people than an honest, challenging problem, whose possible solution will bestow fame and remain as a lasting monument. Following the example set by Pascal, Fermat, etc., I hope to gain the gratitude of the whole scientific community by placing before the finest mathematicians of our time a problem which will test their methods and the strength of their intellect. If someone communicates to me the solution of the proposed problem, I shall publicly declare him worthy of praise.*

The problem he posed was the following:-

> *Given two points A and B in a vertical plane, what is the curve traced out by a point acted on only by gravity, which starts at A and reaches B in the shortest time.*

In this project, I plan on deriving a solution to this problem by gradient descent.By, this new method of solving, we will be able to find a path whose time taken will be as close as possible to any nearest time value.To, elaborate, if we want to find a path in the vertical plane such that the time taken by a frictionless bead travelling on the required path is exactly 1.75 times the time taken by the same frictionless bead travelling on a straight line, we can do that. All this is made possible by using gradient descent, backbone and primary working principle of all types of Neural Networks.

This new method of a traversing a function in a given domain is very powerful, and it allows us to arrive at a solution without using integration, i.e. we can arrive at a numerical solution for any path for any given amount of time taken, without ever having to integrate.Another way this method is very advantageous is it can quite easily take care of finding optimal solutions upon functions which are not integrable or are quite hard to integrate.

Before I use the numerical gradient descent method to solve the Bracistochrone problem, I will first solve the problem analytically, i.e. using Integration and the Euler-Lagrange Equation.The theoritical solution thus obtained is the cycloidal curve connecting the 2 points.

(*) The time taken to travel from pt P1 to point P2 is given by the integral:

$$t_{1\to 2} = \int_{P_1}^{P_2} \frac{ds}{v}$$; ds - is the distance

v - is the average velocity of the body.

Given that there is no friction and only external force on the system is gravity,

By conservation of Energy, we have,

$$\frac{1}{2}mv^2 = mgy$$; y is the height

$$\Rightarrow v = \sqrt{2gy}$$

$$ds = \sqrt{dx^2+dy^2} = \sqrt{1+y'^2}\,dx$$

$$\Rightarrow t_{1\to 2} = \int_{P_1}^{P_2} \frac{\sqrt{1+y'^2}}{\sqrt{2gy}}\,dx \Rightarrow t_{1\to 2} = \int_{P_1}^{P_2} \sqrt{\frac{1+y'^2}{2gy}}\,dx$$

$$\Rightarrow f = (1+y')^{1/2}(2gy)^{-1/2}$$

Applying E-L Eqn: $\frac{\partial f}{\partial y} - \frac{d}{dx}\left(\frac{\partial f}{\partial y'}\right) = 0 \Rightarrow \boxed{f - y'\frac{\partial f}{\partial y'} = C}$ ← constant

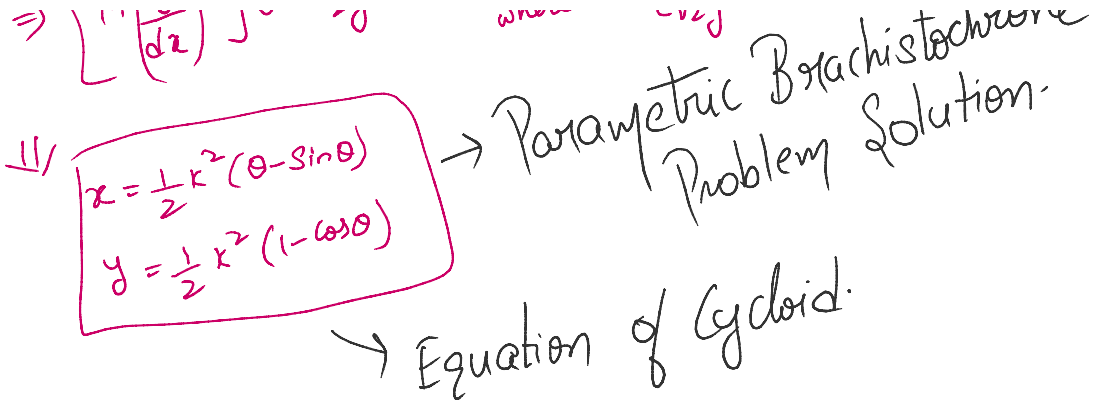
$$\Rightarrow \frac{\partial f}{\partial y'} = y'(1+y'^2)^{-1/2}(2gy)^{-1/2}$$

$$\Rightarrow \frac{1}{\sqrt{2gy}} * \frac{1}{\sqrt{1+y'^2}} = C$$

$$\Rightarrow \left[1+\left(\frac{dy}{dx}\right)^2\right]y = \frac{1}{2gC^2} = K^2$$ → New constant

where $K = \frac{1}{C\sqrt{2g}}$

...tric Brachistochrone ...tion.



So, above, I have shown the parametric solution to the brachistochrone problem obtained from the conventional mathematics.

The new method that I present in this project involves gradient descent, which is a way to navigate around in a differentiable function. In a nut shell, what gradient descent allows us to do is navigate around in a function in a intelligent manner, so that you are able to arrive at any value you want to on a function. If the value we want to go to is not available as an output of the function, i.e. the value we want is out of the range of the function, then we will arrive at a value of the function, that is closest to the given value.

In our case, the function we want to navigate around is the time taken by a ball to move along a given path under the influence of just gravity.Any curve on the XY plane is characterised by the y-values it goes to for each x-value.

In this method, I divide the X- axis between start point and end point into 50 or 100 points of equal distance amongst them, and then Y values for each point in x-axis is arrived at or calculated so that the time of travel is minimal.

So, we have 50 very small straight lines in a very small x axis gap of 1 unit and y axis gap of 0.5 units.i.e. my initial and final points are: (0,0.5),(1,0).Note, if we want to find the path between points that are very distant say between (100,100) and (200,0), then we will need to take more than 50 lines attaching in different angles between them, i.e. we will need to divide the x axis into more than 50 points, say 5000 or higher, to obtain a good resolution and approximation of the curve into many small lines. The point is that the curve will be better approximated only if we have a huge number of very small straight lines as compared to the distance to be traversed.

Now, that we have setup the optimisation problem, it is now time to define the variables and function.The variables here are the yvalues for each value of x.The time taken by a body to go down a path is decided by these y values. The total time taken will be the sum of times taken by a body to go from x1,y1 to x2,y2 and x2,y2 to x3,y3 and x3,y3 to x4,y4 and so on.

X1,y1 to x2,y2 is a straight line and then x2,y2 to x3,y3 is a straight line and so on. Due to conservation of energy, the velocity of the body at a position y is given by :

$$v_{y_k} = \sqrt{2g(y_i - y_k)} \quad \rightarrow \text{Due to conservation of Energy.}$$

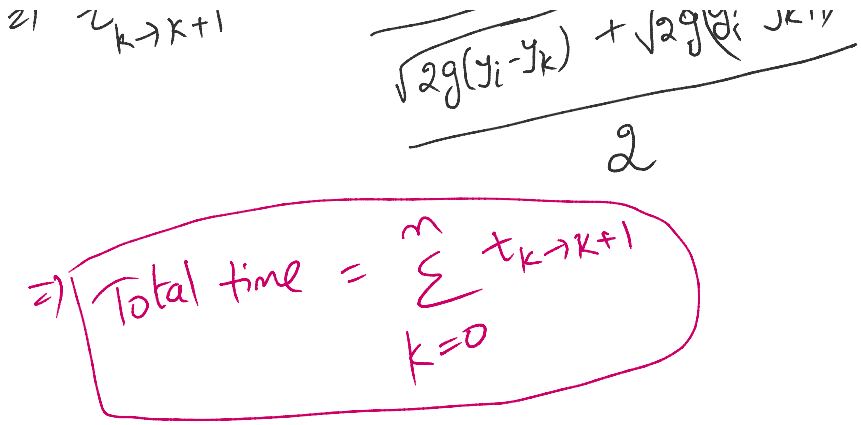
y_i - is Starting y-coordinate.

time taken to go from (x_k, y_k) to $(x_{k+1}, y_{k+1}) = \frac{ds}{v_{avg}}$

$$v_{avg} \approx \frac{v_{y_k} + v_{y_{k+1}}}{2} = \frac{\sqrt{2g(y_i - y_k)} + \sqrt{2g(y_i - y_{k+1})}}{2}$$

$$ds = \sqrt{(x_k - x_{k+1})^2 + (y_k - y_{k+1})^2}$$

$$\Rightarrow t_{k \to k+1} = \frac{\sqrt{(x_k - x_{k+1})^2 + (y_k - y_{k+1})^2}}{\frac{\sqrt{2g(y_i - y_k)} + \sqrt{2g(y_i - y_{k+1})}}{2}}$$



The above method is used to calculate the total time taken by the body to go from initial fixed point to final fixed point via a curve, which is characterized by the yk's. Thus we now have a target function (i.e. the time taken for the whole journey) as a function of variables yk's, where k goes from 0 to 50(in our case, sometimes, 1 unit length is divided into 100 points for better resolution obtainment). Now, all gradient descent does is find the derivative of the total time function wrt each yk, i.e. for each small change in yk, we have change in total time of traversal and the ratio of change in total time to change in that particular yk is taken as gradient of time wrt yk. This calculation of gradient is done for all 50 of the yks and for each iteration, the gradient of time function wrt yk, multiplied by a learning rate, is substracted from initial yk. This process is done for all yk's within one iteration.Now, the time function and gradients are all calculated wrt the new yks and the new gradients after multiplication with a learning rate is substracted from yks (obtained at the end of the 1st iteration), to obtain yks after the end of second iteration. This keeps on happening till we stop. In our case, I had to go to 20000 iterations, to finally come about the minimal value of the time function and find the corresponding yks that create this value of total time of traversal.

The learning rate, number of iterations are all hyper parameters, which vary on the basis of the tolerable error, the range of X,Y values and the number of yks themselves.

The above said procedure is implemented in code to obtain perfect cycloid in brachistochrone.ipynb.

This above procedure, known as gradient descent arrives successfully at the minimal value of the function. Now, say we want to find out the path that has not minimum time of traversal, rather the path that takes a specific amount of time, then after calculating the total time of traversal, we need to find mean squared error between the found time and the time required. This mean squared error should then be used for gradient descent, i.e. this mean squared error should be used for calculating yk's gradients and their subsequent updating. This leads to the gradient descent minimizing the mean squared error between total time of traversal and the time required, rather than the total time of traversal as was done in the last case, this leads to the gradient descent minimization of the mean squared error between total time of traversal and required time. The yks obtained at the minimal value of the error is the path that leads to a body descending with time of traversal as close to the time required as possible. This closeness is characterized by the minimal value of the mean squared error. The smaller the final mean squared error, the closer is the path thus obtained is nearer to the required path(i.e. the one that takes exactly treq amount of time).

This finding different paths having times of traversal nearer to different times required is shown and derived in code in brachistochronesimulation.ipynb