Brachi

Group 4

Sonny Cervienka Glyn Finck Daymon Krotez Jake Wickstrom Aidan Wilson

May 24, 2018

1 Introduction

This document outlines the fundamentals of "Brachi", a web based physics simulation/game which allows users to predict and compare the optimal path of a marble in an arbitrary two-dimensional potential energy field, similar to the Brachisticrone problem. Here, 'optimal' refers to the path which minimizes the time taken for the marble to traverse the curve.

All of the code referenced in this paper can be found at the project GitHub page:
https://github.com/jake-wickstrom/brachi

2 Path Optimizer

2.1 Derivation of Equations

Calculus of variations was used to find the optimal trajectory of the marble. Here, 'optimal' refers to the path that minimizes the time taken for the marble to traverse the curve. Using the stated method yields the following functional to be optimized:

$$\tau\Big[\vec{s}(t), \dot{\vec{s}}(t), U\big(\vec{s}(t)\big)\Big] = \int_{t_0}^{t_f} dt \tag{1}$$

Where τ is the time taken to go from the start point to the end point, U is the 2D potential energy field, and $\vec{s}(t)$ is the position of the marble. By parameterizing the curve in Cartesian coordinates in terms of arc length s we get:

$$\tau\Big[x(s), y(s), \dot{x}(s), \dot{y}(s), U\big(x(s), y(s)\big)\Big] = \int_{s_0}^{s_f} \frac{ds}{v} \tag{2}$$

Where $v = \frac{ds}{dt}$ is the speed of the marble. From the definition of translational kinetic energy $T = \frac{1}{2}mv^2$ and conservation of energy we have:

$$v = \sqrt{\frac{2}{m}(E - U)} \tag{3}$$

Where E is the total energy of the marble and m is its mass. From the 2D Cartesian metric we have:

$$ds = \sqrt{dx^2 + dy^2} = \sqrt{\left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2} ds = \sqrt{\dot{x}^2 + \dot{y}^2} ds \tag{4}$$

Where $\dot{x}, \dot{y}$ represent derivatives with respect to arc length s (ie. $\dot{x} = \frac{dx}{ds}$). Thus, combining (2), (3), and (4) and scaling the arc length conveniently yields:

$$\tau = \int_0^1 \sqrt{\frac{\dot{x}^2 + \dot{y}^2}{\frac{2}{m}(E - U)}} ds \tag{5}$$

This functional can be optimized using the Euler-Lagrange equation, with:

$$\mathcal{L} = \sqrt{\frac{\dot{x}^2 + \dot{y}^2}{(E - U)}} \tag{6}$$

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{d}{ds}\frac{\partial \mathcal{L}}{\partial \dot{x}}, \ \frac{\partial \mathcal{L}}{\partial y} = \frac{d}{ds}\frac{\partial \mathcal{L}}{\partial \dot{y}} \tag{7}$$

Evaluating the Euler-Lagrange equation (7) for x, noting that U is itself a function of x, y, and ultimately s, gives:

$$\begin{aligned}
\frac{U_x(\dot{x}^2 + \dot{y}^2)^{1/2}}{2(E-U)^{3/2}} &= \frac{d}{ds}\frac{\dot{x}}{\big((E-U)(\dot{x}^2 + \dot{y}^2)\big)^{1/2}} \\
&= \frac{\ddot{x}}{\big((E-U)(\dot{x}^2 + \dot{y}^2)\big)^{1/2}} - \frac{\dot{x}\frac{d}{ds}\big[(E-U)(\dot{x}^2 + \dot{y}^2)\big]}{2\big((E-U)(\dot{x}^2 + \dot{y}^2)\big)^{3/2}} \\
&= \frac{\ddot{x}}{\big((E-U)(\dot{x}^2 + \dot{y}^2)\big)^{1/2}} - \frac{-\dot{x}\frac{dU}{ds}(\dot{x}^2 + \dot{y}^2) + (E-U)(2\dot{x}\ddot{x} + 2\dot{y}\ddot{y})}{2\big((E-U)(\dot{x}^2 + \dot{y}^2)\big)^{3/2}} \\
&= \frac{\ddot{x}}{\big((E-U)(\dot{x}^2 + \dot{y}^2)\big)^{1/2}} + \frac{\dot{x}(\dot{x}U_x + \dot{y}U_y)(\dot{x}^2 + \dot{y}^2) - (E-U)(2\dot{x}\ddot{x} + 2\dot{y}\ddot{y})}{2\big((E-U)(\dot{x}^2 + \dot{y}^2)\big)^{3/2}}
\end{aligned} \tag{8}$$

By the chain and product rules. Simplifying and collecting $\ddot{x}$ terms this gives:

$$\ddot{x} = \frac{U_x(\dot{x}^2\dot{y} + \dot{y}^3) - U_y(\dot{x}^3 + \dot{x}\dot{y}^2) + (E-U)\dot{x}\ddot{y}}{2(E-U)\dot{y}} \tag{9}$$

And we can clearly see by symmetry of $\mathcal{L}$ that the Euler-Lagrange equation for y yields:

$$\ddot{y} = \frac{-U_x(\dot{x}^2\dot{y} + \dot{y}^3) + U_y(\dot{x}^3 + \dot{x}\dot{y}^2) + (E-U)\dot{y}\ddot{x}}{2(E-U)\dot{x}} \tag{10}$$

This system of differential equations has boundary conditions:

$$\begin{aligned} x(0) = x_0,\ y(0) = y_0 \\ x(1) = x_f,\ y(1) = y_f \end{aligned} \tag{11}$$

It should be noted that the Euler-Lagrange equation does not guarantee finding an absolute minimum, only local extrema. For a sufficiently well behaved potential field however, this should not be a major problem.

2.2 Solving the Equations

Equations (9) and (10) are highly coupled nonlinear second order differential equations. Solving the system in general is not possible analytically, particularly as a boundary value problem. After significant research and experimentation with numerical solving methods it was decided to implement the shooting method with the aid of MATLAB's ode15i implicit differential equation solver. The shooting method involves selecting various initial conditions, solving the system for those conditions, and finding the best possible fit for the desired boundary conditions.

The procedure designed for finding an optimal trajectory is as follows:

1. Evaluate the system of equations for a given potential U.
2. Convert the system to a system of four first-order equations.
3. Solve the system numerically using a given starting point and some initial $\dot{x}$ and $\dot{y}$ conditions.
4. Linearly interpolate the solution to 1000 points.
5. Search for intersections of the interpolated solution and the line formed between the initial and final points.
6. For each intersection found truncate the solution at that point, scale it to fit the initial and final points exactly, and linearly interpolate to 1000 points once again. These are candidate trajectories.
7. Evaluate the time of each candidate trajectory using an energy-based finite difference (the same method used in section 3.2). Reject trajectories which pass through forbidden regions (these yield complex times).
8. Repeat steps 3 to 7 for a chosen set of initial $\dot{x}$ and $\dot{y}$ and store the trajectory with the lowest time.

There are some drawbacks to this method. Interpolation is required because the solution to the system can have large distances between points in some cases. Some type of spline would be ideal, but anything besides linear interpolation is too slow. In some cases the candidate trajectories may only contain a handful of points from the original system solution. This is ultimately due to limitations of the ode15i solver and improving it was beyond the scope of this project. The result of these limitations is that the solutions may not be perfectly optimal. They should, however, be of a form very similar to the optimal solution.

In order for the system be solved numerically it is necessary to have a non-zero initial kinetic energy. If T is initially zero it results in division by zero at the start point ($E = U$) and bad behaviour from the solver. An initial T on the order of 10^{-6} is sufficient to avoid these problems and does not significantly affect the times.

The MATLAB code used for finding solutions can be found on the Brachi GitHub repository at **https://github.com/jake-wickstrom/brachi/path-optimizer/**. "path_optimizer.m" is the main file.

2.3 Found Solutions

Solutions were found for 7 different potential energy fields:

Brachistochrone	$U = y$
Brachistochrone^2	$U = y^2$
Half Pipe	$U = (y-0.5)^2 - x$
Asteroids	$U = -\frac{0.03}{\sqrt{((y-0.5))^2+((x-(.95-.05)/2))^2}} - \frac{0.1}{\sqrt{((y-0.5-0.18*2))^2+((x-.95))^2}} - \frac{0.06}{\sqrt{((y-0.1))^2+((x-.8))^2}} - \frac{0.04}{\sqrt{((y-0.15))^2+((x-.15))^2}}$
Charge Pipe	$U = (y-0.5)^2 - x + \frac{0.1}{\sqrt{(y-0.5)^2+(x-0.45)^2}} - \frac{0.1}{\sqrt{(y-0.86)^2+(x-0.95)^2}}$
Green Hill Zone	$U = \sin(\pi x) + \sin(\pi y)$
Green Hill Zone 2	$U = \sin\big((y+0.5)\cos(2\pi x)\big)$

Visualizations of these potentials and their corresponding solutions can be seen by playing the Brachi game (see section 4).

3 Front End Physics Simulation

3.1 User Interface

The simulation can be accessed at the appropriate URL by any web browser as long as the server is running correctly. Once connected users can navigate through the various options and select the desired potential field. In the main simulation page for each level two circles are shown to indicate the start and end positions for the path to be drawn. As long as the input curve starts and ends within these circles any arbitrary path can be drawn with the cursor. The application will sample the location of the mouse as many times as possible and filter out redundant or problematic points. After being filtered a parametric cubic spline is interpolated between each point to ensure that a smooth continuous curve connects the end points.

All of the front end user interface and simulation code is written in JavaScript available on the Git Hub repository.

3.2 Simulation

The motion of the marble along the input path is simulated by stepping through arc length segments ds of the curve and then calculating the time taken dt for the marble to traverse each segment. Using regular Cartesian coordinates with the y-axis directed upwards and the x-axis to the right the position of the marble along the curve at time t is given by $s(t)$. Taking the time derivative gives the marbles velocity as

$$v = \frac{ds}{dt} \tag{12}$$

Since ds is a straight differential line segment using Pythagoras' Theorem:

$$ds = \sqrt{dx^2 + dy^2} \tag{13}$$

By conservation of energy, the total energy of the marble at the start of the path must remain constant during it's motion (assuming no losses). If the marble starts from rest (ie. no initial kinetic energy) then it's total energy at the start E_{total} depends only on the potential field being used and thus

$$E_{total} = U_0 = T + U \tag{14}$$

Where U_0 is the marble's initial potential energy and U and T are it's instantaneous potential and kinetic energy respectively at any position along the curve. The kinetic energy of the marble is $T = \frac{1}{2}mv^2$ where m is the mass of the marble. Plugging this into (14) and solving for v yields:

$$v = \sqrt{\frac{2}{m}(U_0 - U)} \tag{15}$$

Now combining (12) and (13) into (15) and solving for dt

$$dt = \sqrt{\frac{dx^2 + dy^2}{\frac{2}{m}(U_0 - U)}} \tag{16}$$

This is now all that is needed to simulate the motion of the marble along the path. The simulation steps through each (x, y) point in the parametric spline and calculates dx, dy and the new potential energy. Each time step can then be summed to yield the total time that it took the marble to complete the path.

3.3 Animation

The data structure used to store the trajectory of the marble is three arrays of equal length. One for each x-coordinate, one for each y-coordinate and one for each time step dt ordered from the start of the curve to the end.

The animation iterates through each (x, y) point in the curve and then pauses for the corresponding dt before redrawing the marble at the next point. However, the delays passed into the 'pause' function were of comparable size to the inherent delays of redrawing the marble on the screen. This meant that the speed of the marble appeared to be incorrect.

To fix this a time correction function was implemented that would record the length of time that the animation actually paused for and allow the marble to be redrawn at a point that better corresponded to the total elapsed time.

4 Web Server

This section provides a brief overview on how to set up and run the Brachi server on a computer and access the application via a web browser; it is not necessary in understanding the physics or math of the project.

4.1 Requirements

Before the server can be run the following must be completed:

1. Clone the GitHub repository **https://github.com/jake-wickstrom/brachi**
2. Have Python 3 installed. The project was developed using Python 3.6.4, compatibility with other versions is not guaranteed. Python can be downloaded from: **https://www.python.org/downloads/**

3. Install Django. Django is the Python package that was used to create the web server. The project was developed with Django 2.0.3 and compatibility cannot be guaranteed with other versions. For help installing Django see: **https://docs.djangoproject.com/en/2.0/topics/install/**

4.2 Running the Server Locally

The instructions in this section will allow the Brachi server to be run locally on the desired computer, viewable and playable through a web browser running on the same computer.

1. Open a Command Line (Windows), PowerShell (Windows; alternate), or Terminal (MacOS/Linux) prompt and navigate to the "brachi/django-server/brachi" directory (from the location the repository was cloned to). Typing "ls" or "dir" should display a file named "manage.py". If this is not the case then check that the current directory is correct.
2. If this is the first time that the server is being run on this machine, type "python manage.py migrate" to configure the database. This command will not cause any problems if the server has already been run before, so if in doubt simply enter the command.
3. Type "python manage.py runserver" to run the server. Open a web browser and go to "127.0.0.1:8000/navigation/title" in the address bar to start playing!

As side note, the database for the application is used to store user's times and usernames. If the database ever needs to be reset type: "python manage.py flush"

4.3 Running the Server on a Wi-Fi Node

The instructions in this section allow the server to be accessed by anyone on the same Wi-Fi node as the machine running the server. This section is more involved than the previous and thus make sure to be familiar with the previous steps before proceeding. The steps below may not work on every W-Fi node as each router may be configured differently; the configuration process will not be described here. If unsure of the configuration of a particular router try the below steps regardless as they may work (they often do).

1. Make sure that the host machine is connected to the desired Wi-Fi node.
2. Find the IPv4 address of this Wi-Fi node and make a note of it. This can generally be found in the properties section of the network connection (use Google for operating system specific instructions).
3. In a Command Line, PowerShell, or Terminal window, navigate to the "brachi/django-server/brachi" directory. Type "python manage.py runserver 0.0.0.0:8000" exactly as it is written to run the server on port 8000 of the host machine.
4. Anyone on the same Wi-Fi node (including the host computer) can go to the following url in a web browsers to access the server: "128.189.231.64:8000/navigation/title", (replacing the sample IPv4 address 128.189.231.64 with the IPv4 described in step 1). Have fun!