The Minimal Time Problem Interface

Thomas Ayton, Zachary Bradshaw, Brandon Dellucky, Quoc-Anh Thomas Le, Sijing Liu, Ryan Manuel, Billy Musgraves, George Owen, Emily Ribando-Gros, Pujan Shrestha, Ashley Welborn
Louisiana State University , Math Consultation Clinic, Supervised by Dr. Wolenski

BACKGROUND

The inspiration for this research project comes from a problem proposed by Timothy Pennings thought of when bringing his dog Elvis to a beach. In particular, Pennings had a seemingly ridiculous question: Do dogs know calculus? Make no mistake, it is a ridiculous question, however this thought experiment proved interesting.

Essentially, our project's genesis comes down to an overeager dog on the beach. When Pennings would throw a tennis ball into the water, he noticed that Elvis did not travel in a straight line from the beach to the ball, rather Elvis entered the water at an angle with respect to his path on the beach. Understanding the canine's propensity for enthusiasm, Pennings conjectured that this was an instinctive response on Elvis's part to minimize the time to reach the ball, as opposed to minimizing the distance. It was obvious that Elvis could run in the sand much faster than he could swim in water, thus the optimal route would aim to spend more time on the beach then in the water.

Pennings ultimately ran some trials to conclude Elvis was indeed doing this. Despite this statistical evidence, Pennings determined Elvis did not know calculus admitting that, "In fact, he has trouble differentiating even simple polynomials." Still, how could Elvis learn calculus if he, as a dog, is under the constant temptation to eat his homework? Timothy Pennings's question has manifested itself in the form of software that this class has developed.

PROBLEM FORMULATION

Given two regions with two different speeds, how might we minimize the time to travel from a point in the first region to a point in the second region? More generally, can we construct the set of all reachable points given a fixed time, the so called reachable set? Or mathematically:

Consider two open half spaces $M_1$and M_2 with an interface separating the two regions, Σ. This means there is a vector $\vec{n} \neq 0$ and a number $r \in \mathbb{R}$ so that

$$M_1 = \{x: \langle \vec{n}, x \rangle < r\} \text{ and } M_2 = \{x: \langle \vec{n}, x \rangle > r\}$$

and the interface is

$$\Sigma = \{x: \langle \vec{n}, x \rangle = r\}$$

Each M_i, $i = \{1, 2\}$, has an associated velocity set F_i. In particular, let F_i be convex and bounded with $0 \in int(F_i)$. Now identify two points, $X_1 \in M_1$ and $X_2 \in M_2$. Then the minimal time problem finds the trajectory from X_1 to X_2, using the velocities in F_1 while in M_1 and F_2 while in M_2, which takes the least amount of time. Another way of phrasing this problem is to find the point $Q \in \Sigma$ so that together the time it takes to travel from $X_1$to Q and Q to X_2 is minimized.

We considered three cases: a trajectory completely within M_1, a trajectory from M_1 into M_2, and a trajectory from M_1 then along Σ and finally back into M_1 (so called "Swim Run Swim" or SRS). The SRS path is only constructed if for velocities chosen, $v_i \in F_i$, $v_1 < v_2$. Furthermore, the SRS trajectories would not be reachable with a trajectory completely contained in M_1, since we are using a faster velocity on Σ. By discretizing the angle around the starting point, we were able to construct trajectories in several directions. If the trajectory intersected the interface at any point, the law of refraction (also known as Snell's Law) was applied.

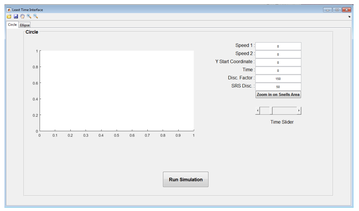
In some cases, it was necessary to calculate the Brewster angle. This is the angle at which light begins to reflect rather than refract in Optics. For a trajectory precisely incident to Σ at the Brewster angle, the outgoing trajectory travels along Σ. It is in this case that we construct the SRS paths.

GUI IMPLEMENTATION

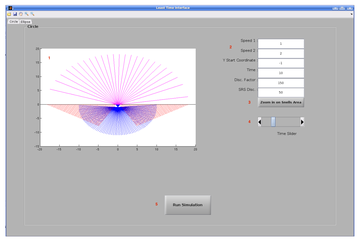
The following figures show the general layout and functionality of the implemented GUI.

What does the GUI do?

Given the initial inputs, including the speed in the first and second regions, starting coordinate in the y direction, and total time, the GUI calculates and plots the reachable set. There are two different tabs for the circular and elliptical velocity sets.

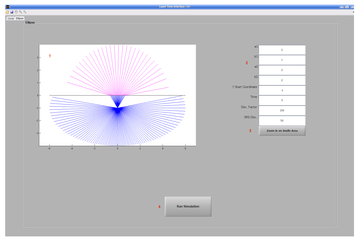


This figure represents the basic layout for the circle.



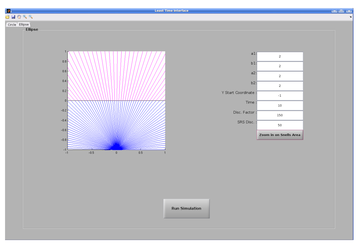
Reachable set for circular velocity set

1. Graphical window
2. Text fields
3. Zoom
4. Time Slider
5. Run Button

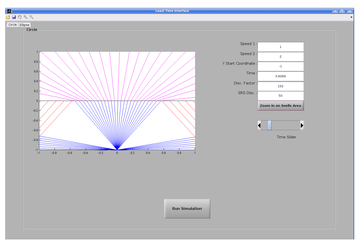


Reachable set for elliptical velocity set

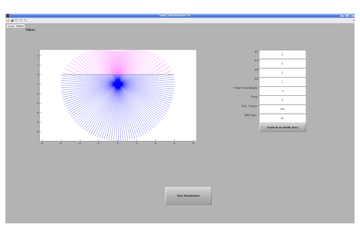
1. Graphical window
2. Text fields
3. Zoom
4. Run Button



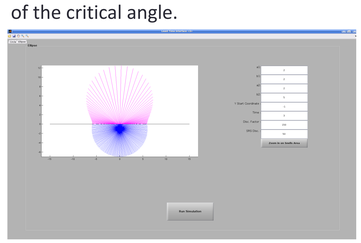
Intersection at the interface for elliptical velocity set



Intersection at the interface for circular velocity set. We have used the zoom button to more clearly show the effect of the critical angle.



This example includes a circular velocity set in the first region (blue) and an elliptical one in the second region (pink). The velocities are chosen such that there is no critical angle.



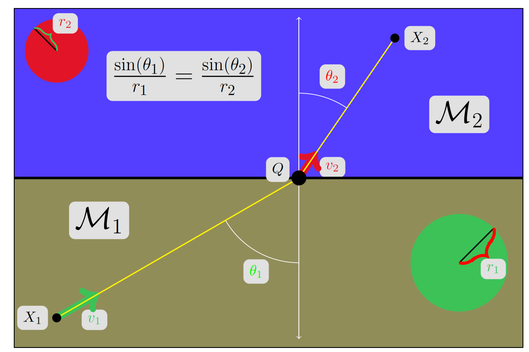
This example includes a circular velocity set in the first region (blue) and an elliptical one in the second region (pink) with a larger vertical velocity in comparison to the figure on the left. The velocities are chosen such that there is no critical angle.

RESULTS, DISCUSSION, AND FUTURE WORKS

So far we have constructed the reachable set for any circular velocity set. We leave it to another group to find the non-optimal points in the first region. That is, the points reachable by a path strictly in the first region which are more rapidly reached by taking an SRS path. While it is certainly an interesting point, this group was primarily concerned with the construction of the reachable set induced by different velocity sets, as well as a new convex optimization approach to the least time problem.

Additionally, we began work on a second tab of our GUI that constructs the reachable set for elliptical velocity sets. This case is easily seen in the two figures at the bottom of the second column. In an elliptical velocity set, the speed at which the trajectory moves is dependent on the angle from the x-axis (we could just as well set this axis anywhere else so long as we are consistent). As it turns out, the optimal trajectory for any given angle is then calculated using Snell's Law as in the case of the circular velocity set. Whenever a trajectory intersects the interface, we apply Snell's law to induce a refraction. Like the circular velocity sets, there are evidently non-optimal points in the first region given an elliptical velocity set, which we leave to a future group to obtain.

In the future, one might attempt to add a third region to the GUI first for the easy circular velocity sets and then for more irregular shapes. It is tempting to then add more regions, but the computational complexity of the algorithm would very quickly need to be addressed. For each region added, we must keep track of another interface and whether or not it is being intersected by any of our trajectories. If so, we must apply Snell's Law for perhaps a second time or a third and so on, among other calculations.



This figure shows an individual case of Snell's law being applied at an interface joining two regions.

ACKNOWLEDGMENT & REFERENCES

We thank Dr. Peter Wolenski for continuously providing undergraduates with valuable research experiences.

References:

1. Pennings, Timothy. "Do Dogs Know Calculus?" The College Mathematics Journal. May 2003.
2. Minton, Roland and Timothy Pennings. "Do Dogs Know Bifurcations?"
3. Perruchet, Pierre and Jorge Gallego. "Do Dogs Know Related Rates Rather than Optimization?" The College Mathematics Journal. January 2006.