# Optimal Mario Kart

Camille Carter, Erik Hannesson, Hayden Parkinson,
Seth Ringger, Ethan Walker

April 10, 2020

## Abstract

Finding the optimal racing controls for the game Mario-Kart and automating the controls would entirely take the fun out of it. However, we thought that this would work as a great project to test our knowledge of optimal control. Mario-Kart consists of system that mimics physics where a racing cart with mass is controlled by steering and acceleration, as well as environment mechanics including drag on different surfaces. We approached the problem with both the HJB equations and Pontryagin's Maximum Principle. Although we were not able to implement an optimal solution in a simplified emulator of the game, we were able to successfully model the system in mathematical equations.

# Background

## Video Game Engines

Every video game must have a setting, the way the world in which the game takes place is defined. Though the plot setting is important, equally important in a video game is the gameplay setting, the way that it feels to play the game and how the game reacts to a player's inputs. The video game engine is what creates this gameplay setting. The engine is a large piece of software that is thought of as the base of the game. The engine defines the rules of the world, how fast a player can run or fall, what can be interacted with, and what the characters look like when they move. [1]
In this project, an important first step was to think about the game engine present in Mario Kart. The way that the game handles movement, input, and motion, also called the game physics, will greatly influence the complexity of our equations and, of course, ultimately the shape of our final solution.

# Mathematical Description

As is typical with racing games, the objective of *Mario Kart* is to complete the course as quickly as possible. We seek the optimal controls of an in-game cart that will do just this – minimize the time taken to finish a race. We will consider the finish line as a fixed point[2] and thus our problem can be classified as a free-time, fixed-endpoint problem.

## Control System

### State Space

In order to adequately describe the current state of our cart, we require three main bits of information:

(i) The cart's position;

(ii) The cart's current velocity;

(iii) The surface the cart is currently on.[3]

It should also be noted that different carts have varying acceleration, maximal velocity, and turning capabilities. Given a fixed cart, these additional characteristics are constant[4] and so we do not explicitly include them in the description of the state space. Instead, they are found in the equations as constants.

[1] The video game engine determines what type of play a player can experience, meaning that the game's genre and story are deeply influenced by the game engine. Think about how there are many games that you can play on a chess board (chess, checkers, and their many varieties) however you could never play pandemic with only a chess board.

[2] Or, more precisely, we will consider a fixed point *on* the finish line.

[3] This can alter the cart's ability to turn and accelerate.

[4] With the exception of varying terrain, in which case they are scaled by some constant factor.

**Control Space**

We restrict our study to the two most basic controls: acceleration and turning. The game allows you to accelerate as well as apply the breaks and reverse, so we consider acceleration as continuous control over full braking/reverse to full acceleration, for the sake of simplicity in the analysis. Turning also has a continuous range, which we assume to span from turning completely left[5] to completely right.

We denote the acceleration and turning controls as $u_1 : [0, \infty) \to [-1, 1]$ and $u_2 : [0, \infty) \to [-1, 1]$, respectively. In the codomain of $u_2$, we identify $-1$ with turning completely left and $+1$ with turning completely right. Taken together, our control equation $\mathbf{u} : [0, \infty) \to [-1, 1] \times [-1, 1]$ is given by

$$\mathbf{u}(t) = (u_1(t),\, u_2(t))\,. \tag{1}$$

# Mathematical Representation

The system we are optimizing has a few variables of which we need to keep track. We will want to know the kart's position at a given time, it's velocity, as well as the direction it is facing. So, we set up the following ODE system with controls

$$\dot{x} = c_1 v \cos(\theta) \tag{2}$$

$$\dot{y} = c_2 v \sin(\theta) \tag{3}$$

$$\dot{v} = c_3 \left( u_1 - \frac{v}{v_{\max}(x, y)} \right) \tag{4}$$

$$\dot{\theta} = c_4 \frac{u_2}{1 + v}, \tag{5}$$

with cost

$$J(u) = \int_{t_0}^{t_f} 1 \; dt, \tag{6}$$

and where

$$v_{\max}(x, y) = \begin{cases} 1 & \text{if } (x, y) \in T \\ 4 & \text{if } (x, y) \in G \\ 1000 & \text{if } (x, y) \in O \end{cases}. \tag{7}$$

$T$ represents the track area, $G$ is the grass area, and $O$ is the out of bounds area. This encourages finding a path that is mostly, if not entirely, on the track

[5] i.e. 90° to the left of forward motion

and certainly not out of bounds by greatly limiting speed off of the track. This system is also dependent on the following constraints

$$u_1 \in [-1, 1] \tag{8}$$
$$u_2 \in [-1, 1] \tag{9}$$

and boundary conditions

$$\begin{aligned} x(0) &= x_0 \\ y(0) &= y_0 \\ x(t_f) &= x_1 \\ y(t_f) &= y_1 \\ v(0) &= 0 \\ \theta(0) &= \theta_0 \end{aligned}$$

$c_1, c_2, c_3$, and $c_4$ are known constants we add in to adjust the motion of our car in the simulation to match the feel of actually playing Mario Kart. We chose $c_1 = 1, c_2 = 1, c_3 = .1$, and $c_4 = .05$.

# Solution

## Setting up the problem for Pontryagin's Maximum Principle

We begin formulating our solution by rewriting our constraints $u_1, u_2 \in [-1, 1]$ as

$$\begin{aligned} u_1^2 - 1 &<= 0 \\ u_2^2 - 1 &<= 0 \end{aligned}$$

and defining our modified Lagrangian

$$\tilde{L} = 1 + \mu_1 \left(u_1^2 - 1\right) + \mu_2 \left(u_2^2 - 1\right). \tag{10}$$

We now define our Hamiltonian

$$H(z(t), u(t), p(t), t) = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \cdot \begin{bmatrix} c_1 v \cos(\theta) \\ c_2 v \sin(\theta) \\ c_3 \left(u_1 - \frac{v}{v_{\max}(x,y)}\right) \\ c_4 \frac{u_2}{1+v} \end{bmatrix} - 1 - \mu_1 \left(u_1^2 - 1\right) - \mu_2 \left(u_2^2 - 1\right)$$

where $z(t) = \begin{bmatrix} x(t) \\ y(t) \\ v(t) \\ \theta(t) \end{bmatrix}$, and with costate equations given by

$$\begin{aligned}
\dot{p_1} &= -H_x = \frac{d}{dx}\frac{c_3 p_3 v}{v_{\max}(x,y)} \\
\dot{p_2} &= -H_y = \frac{d}{dy}\frac{c_3 p_3 v}{v_{\max}(x,y)} \\
\dot{p_3} &= -H_v = -c_1 p_1 \cos(\theta) - c_2 p_2 \sin(\theta) + c_3 \frac{p_3}{v_{\max}(x,y)} + \frac{p_4 c_4 u_2}{(1+v)^2} \\
\dot{p_4} &= -H_\theta = c_1 p_1 v \sin(\theta) - c_2 p_2 v \cos(\theta)
\end{aligned}$$

subject to

$$p_3(t_f) = p_4(t_f) = 0.$$

Note that we have these endpoint conditions on $p_3$ and $p_4$ from Pontryagin's Maximum Principle because there are no right boundary conditions on the corresponding states $v$ and $\theta$. Additionally, the Maximum Principle tells us that

$$H(z^*(t), u^*(t), p^*(t), t) = \max_{u_1, u_2 \in [-1,1]} H(z(t), u(t), p(t), t) \tag{11}$$
$$H(z^*(t), u^*(t), p^*(t), t) = 0 \tag{12}$$
$$\mu_1 \left(u_1^2 - 1\right) = 0 \tag{13}$$
$$\mu_2 \left(u_2^2 - 1\right) = 0 \tag{14}$$
$$\mu_1(t), \mu_2(t) >= 0 \tag{15}$$

for all $t \in [0, t_f]$.

With the equations we get from the Maximum Principle, along with our state equations from above, we now have the tools to solve our system.

We begin by using the condition that the Hamiltonian will be maximized along the optimal trajectory $u_1^*, u_2^*$ (equation 11). We note that $u_1$ shows up in two places in the Hamiltonian: in the terms $p_3 c_3 u_1$ and $\mu_1 \left(u_1^2 - 1\right)$. This second term will always be 0 by condition (13) above. Thus by inspection we determine that the Hamiltonian will be maximized with respect to $u_1$ when

$$u_1 = \begin{cases} 1 & \text{if } p_3 c_3 > 0 \\ -1 & \text{if } p_3 c_3 < 0 \\ ? & \text{if } p_3 c_3 = 0. \end{cases}$$

A very similar analysis tells us that the Hamiltonian is maximized with

respect to $u_2$ when

$$u_2 = \begin{cases} 1 & \text{if } \frac{p_4 c_4}{1+v} > 0 \\ -1 & \text{if } \frac{p_4 c_4}{1+v} < 0 \\ ? & \text{if } \frac{p_4 c_4}{1+v} = 0. \end{cases}.$$

Thus our solution is bang-bang for both $u_1$ and $u_2$.

We think it is unlikely that $p_3$ will equal zero on a non-measure zero interval, and $c_3$ is nonzero, so when we compute the solution numerically we assume that we never fall into the last case where $u_1$ is unknown. We hypothesis the same thing for $u_2$.

At this point, we have a system of eight ODEs from the state and costate equations with eight boundary conditions and eight unknowns, and we opt to solve the system numerically from here.

To put our system into a boundary value problem solver, we need to have a determined end time instead of an arbitrary $t_f$ like we currently have. We make the substitution $s = \frac{t}{t_f}$ to deal with this and note that now our problem is defined over $s \in [0, 1]$. Since $\frac{d}{ds} = t_f \frac{d}{dt}$, the right hand side of each ODE in our system will be multiplied by $t_f$ to complete the rescaling. We solve for $t_f$ by adding it into the state space with the equation $\dot{t_f} = 0$.

Because we have one more unknown from $t_f$, we need one more boundary condition. We use the condition that the Hamiltonian is 0 along the optimal control (equation 12) to get one more boundary condition. Since this true for all $s \in [0, 1]$, we choose to use $s = 0$ for convenience since we have several boundary conditions for our states at $s = 0$.

$$0 = H(z(0), u^*(0), p(0), 0) = |p_3|c_3 + |p_4|c_4 - 1$$

We now solve the system numerically using scipy's solve_bvp function. See plots below for our results.

# Solution Interpretation

## Results from Analysis

Our physical intuition about optimal controls matches our bang-bang result. If you have perfect control of the car, as you do in our model, then it makes sense to reach the extremes of speed and turning to race the car as fast as possible.

Our analysis of the system with the Maximum Principle is also supported by the HJB equation

$$-V_t(t,x) = \inf_{u \in [-1,1]} \left\{ 1 + V_x(v\cos(\theta) + V_y(v\sin(\theta) + V_v \left( u_1 - \frac{v}{v_{\max}(x,y)} \right) + V_\theta \frac{u_2}{1+v} \right\}$$

$$-V_t(t,x) = 1 + V_x(v\cos(\theta)) + V_y(v\sin(\theta)) - V_v \left( \frac{v}{v_{\max}(x,y)} \right) - V_v - \frac{V_\theta}{1+v}.$$

Notice that the infimum in the first equation above will be achieved by $u_1 = -1$ if $V_v > 0$ and $u_2 = 1$ if $V_v < 0$, with corresponding results for $u_2$. If we think about what this derivative means, this makes sense: if the kart is in a state where increasing speed will decrease the time it takes to reach the end, the best thing would be to go as fast as possible. If increasing speed would cause us to take longer to reach the goal (perhaps we would run into a patch of grass), the best thing would be to slam on the brakes to maneuver away. This aligns with our intuition about the optimal control being bang-bang in terms of acceleration.

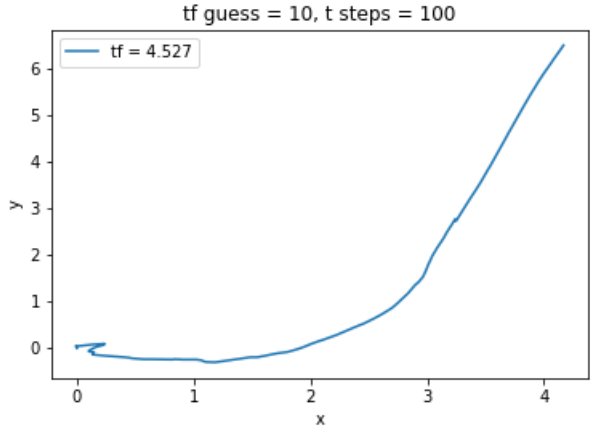
Notice as well that a similar intuition exists for the $u_2$, the control corresponding to the change in the turning angle. A non-zero value for $V_\theta$ implies that there exists a better angle that would get to the end faster. Whenever our driving angle is not optimal, the optimal control is to fix it as fast as possible, meaning $u_2 \in \{1, -1\}$. Only when the angle is already optimal, i.e. $V_\theta = 0$, can $u_2$ take on any other value, but if $\theta$ is already optimal, we know $u_2 = 0$.

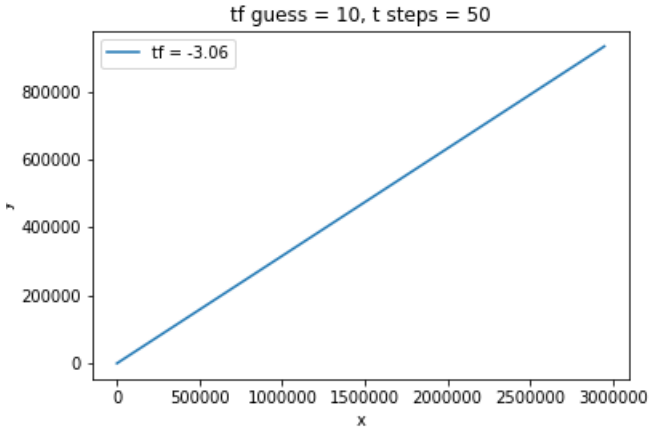
## Results from Numerical Solvers

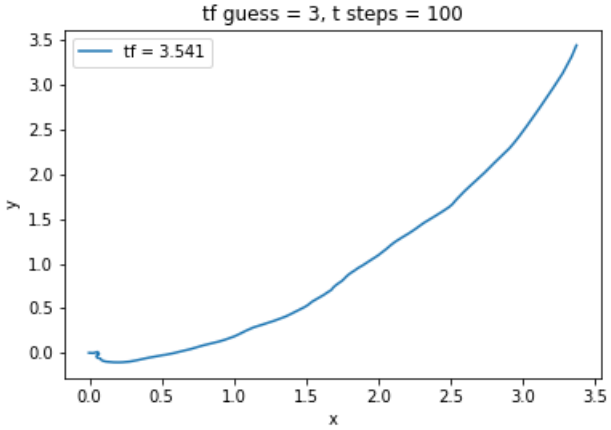
We tried finding similar set-ups in the volume 4 labs, and the closest method was on the river-crossing problem. We attempted to formulate our problem in a similar manner, but with no success due to the river-crossing problem being reducible to a system reliant only on x. Because the max speed of the kart is reliant on x and y independently, this formulation was infeasible. The set-up for that problem was founded on the ability to formulate $\theta$, y and $t$ all in terms of x.

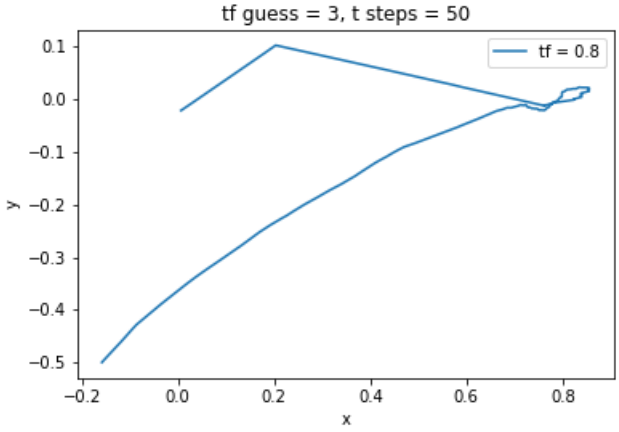
One issue with using the maximum principle to solve the system numerically has to do with what we assume is numerical instability of the system.

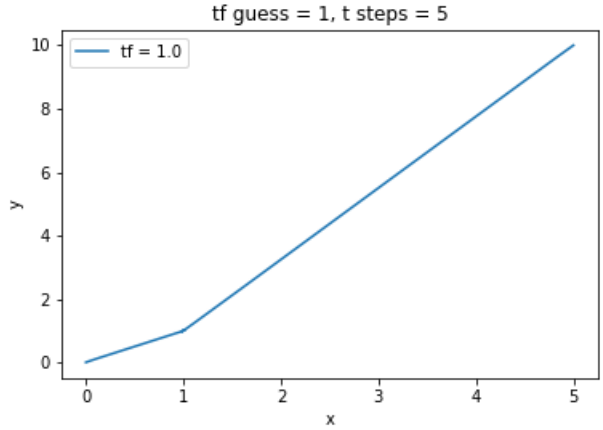
The following are examples of the numerical instability that we encountered. Each graph is a representation of the optimal path that was found by the solver. It is important to note that only one of these example actually ended at the correct point, the final one. As you can see, small changes in the initial conditions and guesses yielded wildly different results. Take notice not only of the shape, but also of the scale.


tf guess = 10, t steps = 100
tf = 4.527
y
x


tf guess = 10, t steps = 50
tf = -3.06
x


tf guess = 3, t steps = 100
tf = 3.541
y
x


tf guess = 3, t steps = 50
tf = 0.8
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
y
−0.2
0.0
0.2
0.4
0.6
0.8
x


tf guess = 1, t steps = 5
tf = 1.0
10
8
6
4
2
0
y
0
1
2
3
4
5
x


NEW RECORD!!
1st 02:09:529 EMW
2nd --:--:--- ---
3rd --:--:--- ---
4th --:--:--- ---
5th --:--:--- ---
BEST LAP 00:12:104 EMW

# Conclusion

One thing that surprised us throughout this project was the difficulty. We began with a problem that we thought would be relatively simple, however as we began to work on it we found that the problem rapidly increased in difficulty, despite the many attempts we made to simplify the systems and problems. We began thinking about the problem in 3-dimensions, since the game is 3-D. However we quickly decided to simplify the problem to a 2-D system. Another simplification we made in the computational portion is related to the shape of the track; all of the tracks in Mario Kart are closed loops. Implementing this reality would greatly restrict the number of feasible curves, complicating the computation. We found that for the sake of computation, it would be simpler to consider a rectangular track, starting in one corner and ending in another.

We are happy that although we made these changes, the only major simplifications were related to the computational solution of the system, so our analysis holds for arbitrary tracks, assuming that one could implement code that represents the track accurately.

Another issue we were not expecting was how numerically unstable the system was. Our experiences with other problems we found in labs were rather different from this, in that they were all tractable problems, easily solved with some simple analysis and code. We learned that the real world is much less forgiving to those that try to understand and describe it, and honestly, that is something that makes it so much more rewarding when we can make real progress to finding a solution, even if that solution is not computationally stable, and very difficult to solve analytically.