Model Predictive Control - EL2700

Finite-Time Optimal Control

Assignment 2

Division of Decision and Control Systems
School of Electrical Engineering and Computer Science
Kungliga Tekniska Högskolan

Samuel Erickson Andersson, Gregorio Marchesini and Mikael Johansson

Figure 1: Charles Leclerc in Monte Carlo

Introduction

Now that you have passed the driving test in Homework 1, you are ready to show your potential behind the wheel. Your first race is in Monte Carlo, one of the most difficult racetracks of the GP tour. Before the race, you're in the paddock with your team to take the last check on your race car and, most importantly, define the raceline on the racetrack that will bring you to glory.

While you are sitting with the engineering team before setting up some computer simulations, you remember your father's words, also a great driver of this tournament:

"Slow is smooth, smooth is fast"

While this seemed counterintuitive at first, you now know that your father was a wise man. "Slow is smooth" refers to the feeling drivers have when they are in full control of their race car. Although they are travelling at a roaring speed, time seems to slow down in the cockpit for the best drivers, such that the driving feels "smooth" and feels like "nobody can catch you". But also, you now focus on the fact that "smooth is fast":

Let's again consider our bicyle model while we are inside a turn. For a given velocity of our vehicle and a given steering angle of our front wheel, the instantaneous turning radius is given by

$$R = \frac{L}{\tan(\delta)}.$$

When we engage in a turn, we know from physics that we always experience a centripetal acceleration pointing toward the center of the turn. The formula for such acceleration is given as

$$a_c = V^2/R$$

where V represents the absolute speed of the vehicle, which we can compute as

$$V = \sqrt{v_x^2 + v_y^2}.$$

At this point, we know that "if there is an acceleration, then there is a force." Such a force can be explained by going slightly beyond the kinematic model we considered in Homework 1. Indeed, the force that causes the centripetal acceleration is the force that the tires of the wheels impart on the vehicle due to the friction with the racetrack. Namely, both the front and rear wheels impart a force, F_{tf} and F_{rt} respectively, on the vehicle every time we turn. So it is this force that allows a car to turn. Now the problem is that the force that the tire can impart is not infinite, but it is limited by the characteristics of the road and the weight of the vehicle (and many other factors that we do not consider here for the moment). Taking a simplifying assumption, we can assume that the maximum total force that the tires can deliver is bounded as

$$F_t \leq \mu \cdot m \cdot g$$

where m is the mass of the vehicle μ is the friction coefficient which is ususally a number between 0.8 and 1.5 depending on the type of tire and the road conditions, while g is the gravitational

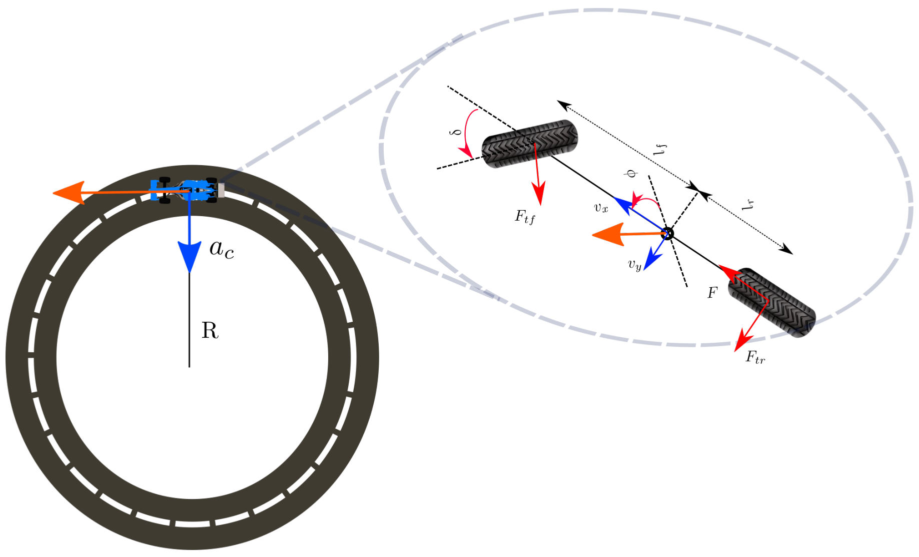

Figure 2: Representation of the centripetal force on the vehicle

acceleration. This means that the maximum velocity we can have for a given turning radius is determined by equating the maximum tire force and the centripetal force as

$$\mu \cdot m \cdot g = mV^2/R$$

from which we derive that

$$V \leq \sqrt{R \cdot \mu \cdot g}.$$

We can thus see that we can go faster for high turning radii (almost straight lines) and slower for small turning radii. Notably, we often reason in terms of curvature instead of turning radius, where the curvature is simply the reciprocal of the turning radius, as

$$\kappa = \frac{1}{R}$$

for which the previous relation becomes

$$V \leq \sqrt{\frac{\mu \cdot g}{\kappa}}.$$

We thus arrived at understanding the good old lesson: we go fast over paths that are almost straight, which are those of minimum curvature and thus "smooth".

Without further ado, it is time we find a raceline that is smooth enough for us to drive fast and win this race!

Software instructions. Before proceeding, let's make sure that you have all the software needed for this assignment. We recommend you to use Ubuntu 20.04 or above, but these instructions can also be completed on Windows and Mac as long as you have a working version of Python 3.10 or above. Before proceeding, install the dependencies as `pip install -r requirements.txt` with Python 3 (unless you are using the virtual machine). For this assignment, we will use the solver MOSEK, which can be used together with CVXPY. First, download an academic MOSEK license at `https://www.mosek.com/license/request/?i=acp`. You should select a personal academic license. You will then receive an email over your institutional email with a file named `mosek.lic`. In order to find the license, the file `mosek.lic` should be placed in a folder called `mosek` in the home directory of your machine. Hence, on Linux/MAC create a folder called `mosek` and place it and save the `mosek.lic` inside of it as

```
$HOME/mosek/mosek.lic
```

On Windows, it will be the same

```
%USERPROFILE%\mosek\mosek.lic
```

Design Task

To complete the task, you will be working with the script `task2.py`, and the Python file `racing_line.py`. We first want to experiment with convex optimization to find the trajectory of minimum curvature and verify that this is indeed faster than just following the centerline. In order to find the centerline, we will divide the track into multiple sections based on the coordinate s. Namely is we let S be the total length in meters of the racetrack, the we can query the position of the centerline for each given curvilinear coordinate s using the class `RaceTrack` as `racetrack.position(s)` and the normal vector to the racetrack as `racetrack.normal_vector(s).` We can then introduce the variable $d(s)$ to denote the displacement from the centerline at s, such that the position of the vehicle for each coordinate s can be defined as

$$p(s) = c(s) + n(s) \cdot d(s),$$

where the condition

$$|d(s)| \leq d_{\text{width}}/2 \tag{1}$$

needs to hold for the car to stay on the track, where d_{width} is the track width. If we consider a finite number of discretization points s_i we can refer to the position of the car at each s_i as

$$p_i = c_i + n_i d_i \tag{2}$$

While minimizing the curvature of a path passing from the points p_i is not possible via convex optimization, we here use the local acceleration as a proxy for the local curvature. Using the second-order difference equation for the acceleration, we have that

$$a_i = \left(\frac{(p_{i+1} - p_i)}{\Delta s} - \frac{(p_i - p_{i-1})}{\Delta s} \right) / \Delta s = \frac{p_{i+1} - 2p_i + p_{i-1}}{(\Delta s)^2} \tag{3}$$

and thus we want to minimize the cost defined by the sum of absolute accelerations

$$\sum_i \|a_i\| \cdot \Delta s. \tag{4}$$

Remember that the optimization variables are the displacements d_i and the vehicle positions p_i, which are just linear functions of d_i as per Equation (2).

Q1: Implement the method `compute_racing_line` in the given Python file `racing_line.py`. The method should compute the racing line with the minimum acceleration that stays within the track.

Q2: Once you have solved the previous problem, you can use the following formula to compute the *actual* curvature of the racing line at an index i:

$$\kappa_i = \frac{|a_{iy} v_{ix} - a_{ix} v_{iy}|}{(v_{ix}^2 + v_{iy}^2)^{3/2}} \tag{5}$$

where

$$v_i = \frac{p_{i+1} - p_i}{\Delta s}. \tag{6}$$

You can then compute the time required to traverse the racing line, assuming that you move at constant speed between the coordinates s_i and s_{i+1} and that you go at the maximum possible speed allowed by your tire's friction model as

$$V_i = \sqrt{\frac{\mu \cdot g}{\kappa_i}}.$$

and thus

$$\Delta t_i = \frac{\Delta s}{V_i}$$

and the total (lap) time is given by the sum

$$T = \sum_i \Delta t_i.$$

Implement the curvature and lap time computation in `compute_lap_time` in `race_line.py`.

Q3: Calculate the racing line for the *Circuit de Monaco* track using your implementations and compare with the racing line defined by the centerline. Is there a noticeable improvement in lap time? When plotted, does the racing line look like you expected it to?

Submission. Your submission should be a .ZIP file, named with your Group number (e.g., `Group1.zip`), containing:

1. a PDF document (can be a scanned document, generated from Word, Latex or others) containing the answers and motivation of the solution to the assignment. Make sure to name your document `Assignment2_STUDENT1_STUDENT2.pdf`;
2. all of the `.py` files with a working task2.py that shows your solution on the plot output.

Good Luck!