Race Line Optimization

DEEPARSHAN KHADKA, Miami University, USA

A program that compares two optimal racing lines generated by the simulation of two cars in a given circuit, to produce a comprehensive graph of the two setups and provide information on which one had the better performance.

CCS Concepts: • **Computer aided design** → **Optimization**; *Data Structure*; Simulation; • **Python** → Programming functions.

Additional Key Words and Phrases: racing line, graph, comparative

1 INTRODUCTION

I have been really into F1 for a while now, and I have always found engineering in racing appealing which is why I wanted to do something that could help a racing team understand their cars better and provide proper input to the drivers. For the next F1 season, it's governing body, the FIA has revealed new generation of cars which claim to have a better aerodynamic build that would reduce turbulent air getting released behind it. This would mean better racing experience on the circuit, but we have not really seen it been driven. So, it is up to the engineers in the racing team to supply all the information they can to the drivers before they can "feel" the car working.

With all the talk being directed at the new hype, drivers and pundits alike have their own view towards the new car. Some drivers have come out to say that going from anything we currently have is a step down, and this is the pinnacle of high speed racing. But the FIA is adamant in claiming that these cars will improve the overall racing experience as they evolution in racing does not always mean increasing speed. The new design sets to bring aerodynamics changes that will make following another car much easier by reducing turbulent air that gets released behind. But how good or bad are the cars then, just on paper?

One of the key factors in succeeding in any motor sport is the ability to follow and stick to the optimal racing line. Being aware of the optimal racing line helps a driver gain crucial time advantage over their opponent. That is why drivers and racing team alike try to get every possible information from a given circuit. Therefore, comparing the best possible solutions of two setup should give us some understanding of how two setups fare against each other.

This problem is one of the most looked at in terms of simulation and optimization, for use in tracks or in racing games. I looked at different ways people have approached this optimization problem. The resources I found, for the most part, shared the same ideas on what "optimal" meant for this scenario. You can either choose the; a. shortest path, b. minimum curvature or c. minimum time. For our optimal end result, we will choose the line that takes the shortest time to go through. [Xiong 2010]

If I can get data on the optimal racing lines for two separate cars, I would then be able to compare each of them and see which setup does what better. I would like to be able to compare two cars of different setups and see how their optimal racing lines would fare against each other in a simulated environment. This would be useful for various implications, one of which has been a highly talked about topic in F1 for the last 5 years; it's the new generation of F1 cars as regulated by the FIA. So, at the least, my program should be able to simulate two cars and produce their trajectory outputs.

Author's address: Deeparshan Khadka, khadkad@miamioh.edu, Miami University, High Street, Oxford, Ohio, USA, 45056.

2 BACKGROUND

There are multiple approaches that I can take to run my desired test; a. would be to build a optimal racing line simulator myself (with basic parameters and complexity), or b. to use an existing in-depth simulator to then work with what it generates out. In my search for a base mathematical function to work out the simulation with, I found a program written by a group of students and teachers at the Institute of Automotive Technology in Munich.

2.1 TUMFTM/global_racetrajectory_optimization

Created under the supervision of Prof. Markus Lienkamp, the global race trajectory optimization is a python program that takes in vehicle and race track data, to simulate a lap on it.[Lienkamp 2021] It runs complex mathematical functions on splines generated through computation on arrays of racetrack and car data. The output is very much depended on combinations of parameter a user specifies. The program uses a lot of functions from existing python libraries such as numPy, MatPlotLib, SciPy and configparser. Furthermore, TUMFTM/trajectory_planning_helpers is highly used in the optimization program, and it can also be found on gitHub, created by the same group. The program is aided by all of this to then output the following, based on parameters supplied:

(1) Trajectory
  (a) Race Trajectory - (default) holds detailed information about the race trajectory.
  (b) LTPL Trajectory - holds source information about the race trajectory as well as information about the track bounds and true reference line (the actual race trajectory has to be calculated based on the stored information).
(2) Output Graphs The program created by TUMFTM is very thorough and computes a lot of crucial graphical information regarding the vehicle on the circuit, plotted with Matlab. Some of which are; the race trajectory, v-a-t vs s, Curvature, Power & Energy consumption, tire forces and much more.

2.2 Simulation

The program takes in the vehicle and racetrack information in the form of csv files and parameter text file. The track information holds the coordinates and dimensions of the track layout that a function can then do the required operation to produce splines of the track's boundaries. Similarly, for a car, it takes the (velocity vs max acceleration, ax_max_machine.csv) and (longitudinal-latitudinal-velocity graph, ggv.csv) to build a function that interacts with the splines. And since much of what the trajectory is going to be like is dependent on specific parameters of the car, the racetrack, and the desired outcome set by the racing team, the parameter text file is of much significance here. So, along with the csv data, the text file also gets parsed into a list that will either regulates the workflow of the optimization call or itself gets fed into a function as some mathematical variable. It uses casADi (complex mathematical functions) and numPy to operate on the arrays of data that interact together to form the simulation.

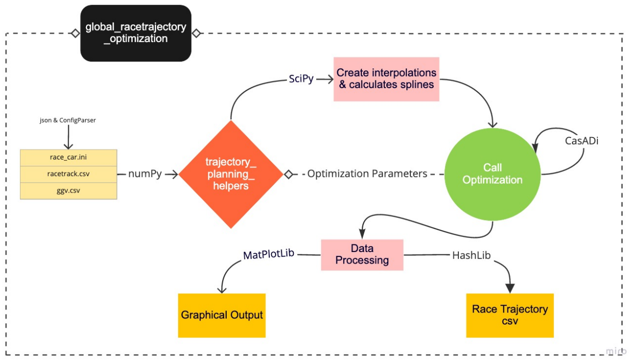


Fig. 1. A detailed flowchart of inputs, outputs and function call of TUMFTM's global race trajectory application.

2.3 System & Software

Ubuntu Virtual Machine running Python application.

3 IDEA DESCRIPTION

Understanding the basic functioning of the software is the key in moving forward with this project. The better my knowledge of the existing program, the better I can pipeline everything together in my implementation.

3.1 Input

Going through the process of understanding their software, I found ways that I could modify the program to my liking. But in doing so, I found out that there were multiple input files that needed to be changed to supply the right data and optimization option. So, first thing I wanted to do was to make a friendlier UI that let modifications to the setting be done without having to visit multiple directories.

Doing so was relatively more difficult than I had anticipated. Since the existing program used configuration parser and json to setup dependency files, I had to learn the library for the ease of appending that particular set of data using those functions. But once everything was set up, my program already became more versatile than what it offered. Now, users are able to chose from these options below, and depending on their choice will get prompted to the required function.

- Run a custom car
- Compare 2 custom cars
- Simulate last run setting

3.2 Optimization Call

Each run on the global_trajectory function takes about 12 minutes per run to output the minimum time solution. So, I wanted to see if I could do something to speed this process up. To do so, I looked more into how the optimization call actually computed all the data. I knew this function only handles minimal mathematical computation to generate specific output and the helper function "trajectory_planning_helpers" handled all the computations on splines. The deeper I dove, the more impressed I was with the software, as it accounted for and computed significant number of variances. Most of the mathematical model consisted of complex differential equations which I could not go through all. So, I gave up the idea of trying to optimize the program and stuck with my final goal of getting a comparative software.

3.3 Output

Since I changed how the input worked, I did not have to go modify individual programs, instead I changed the workflow that got influenced by user's input and supplied new functions that would work with the changed flow. First, I had to make sure the global function took the modified input for my different modes to run. Then to get it to run two setups, I had to create another car configuration array and an output array that stored all critical information required to compare two outputs. Since the existing program produced significant output data, representing all the information was the difficult part. Not all data gets well represented in graphical form, especially comparative as it can easily get messy.

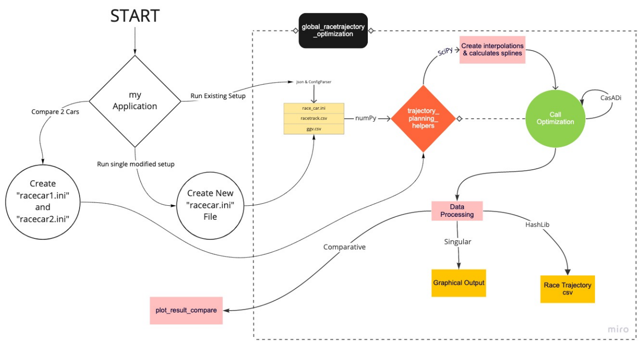


Fig. 2. A detailed flowchart of inputs, outputs and function call of my implementation TUMFTM's global race trajectory application.

3.4 Added Files

All of my need changes were pushed through with these new functions below, and minimal amendments to existing functions.

- Input: myApplication.py, parseOptions.py optionA.py, optionB.py, optionC.py
- Output: main_compare.py, result_plots_mintime_compare, opt_compare.py

4 EXPERIMENTAL SETUP

To run my little experiment, I had to obtain data on the two cars I wanted to get the data on, however it was very difficult as not much is know about the new F1 car, especially to public. So, from whatever I could find, I set up two distinct cars, RedBull F1 Team's 2019 RB16 and a base look of 2022 RedBull car. There were the following differences between the two cars (that could be confirmed):

- **Weight:** New car is 5% heavier (790kgs vs 752kgs)
- **Drag:** New car has reduced drag and better down force (0.7 vs 0.9)
- **Top speed: New car has a very narrow advantage**

**I also did a run comparing RedBull Honda's 2020 car with the Honda Civic to test out the program.*
To do so, I ran my menu, opted to run a comparison between two cars, which then prompted me to change the settings of the two cars. Once all the required parameters were supplied, the program took (on Miami's EGB267 lab computer) 10 minutes for each car, to generate the race trajectory. It ran the 2022 car for 790 iterations while it only took 536

5 RESULTS

The results were positive, and it did show me that different cars will take different paths to get the best racing line for their setup. Even with an increased weight, the new car performed slightly better on the track, due to it's reduced drag rate. Some of the outputs of my program have been posted: Fig. 3, Fig. 4, Fig. 5.

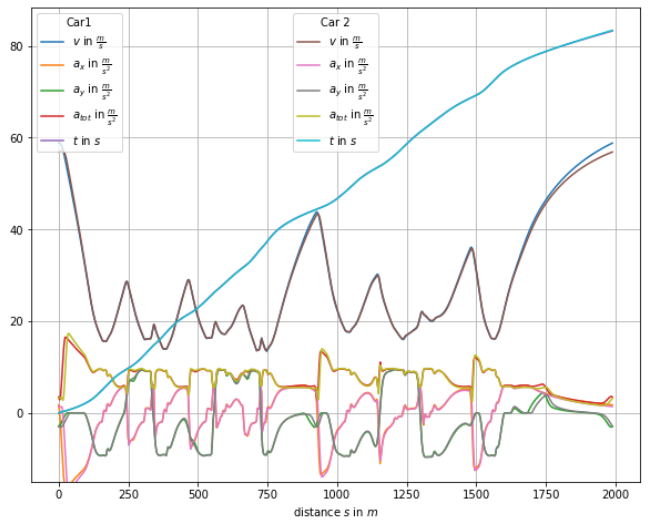


Fig. 3. A comparative graph between Car 1 (2022 F1) vs 2019 RedBull RB16, v-a-t vs s.

Furthermore, after generating the comparative runs, the program then uses the optimal line to generate the time taken. To which, my program compares both the results, outputs the time taken by each, and prints out which one did better (Fig. 6) and the final trajectory it took (Fig. 7).

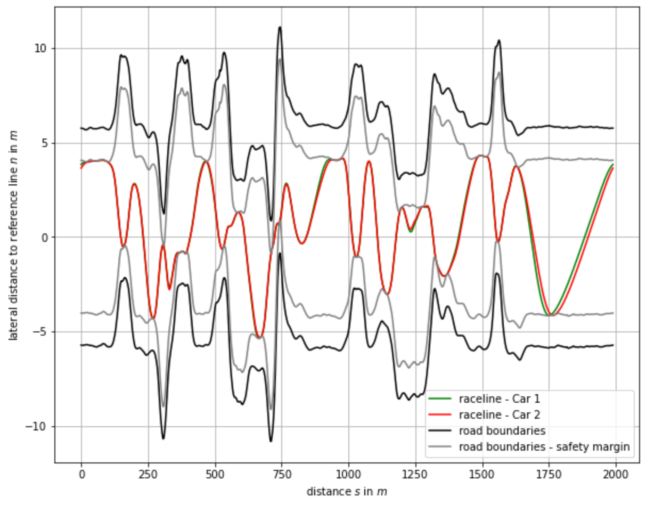


Fig. 4. A comparative graph between Car 1 (2022 F1) vs 2019 RedBull RB16, Race Boundary

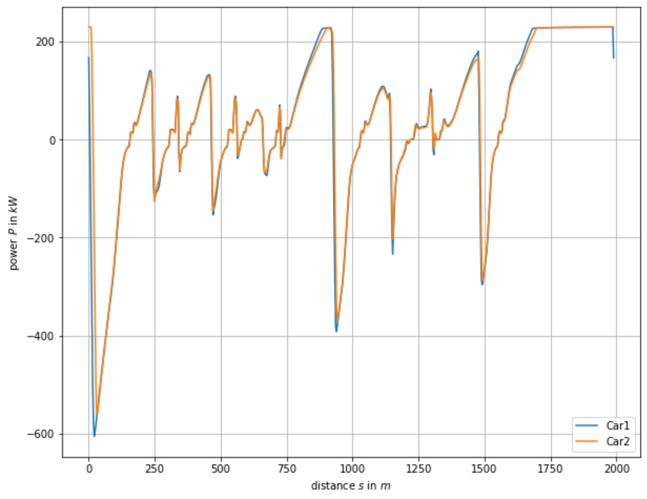
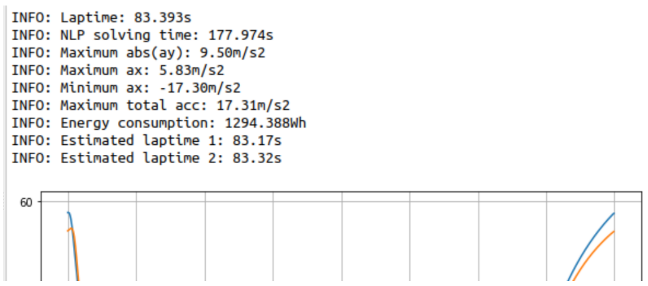


Fig. 5. A comparative graph between Car 1 (2022 F1) vs 2019 RedBull RB16, Power Consumption

As a base benchmark test, I compared 2019 RB16 with Honda Civic Type R, although not completely accurate to the aerodynamics of the Civic. It fared as expected, producing satisfactory results for the program's benchmarks. (Fig. 8)

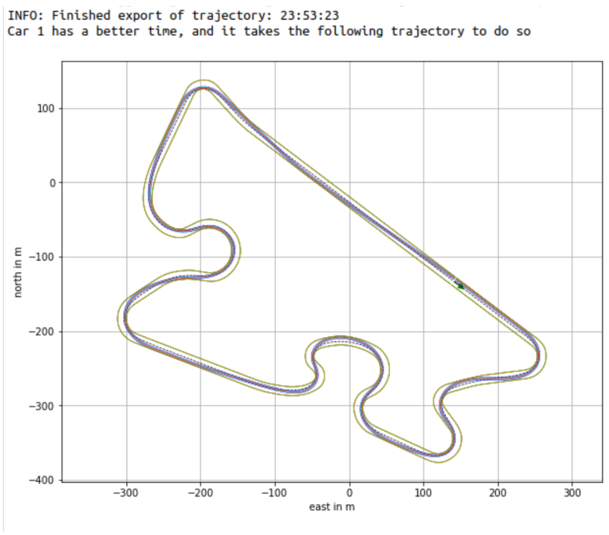
6 CONCLUSION

The program reached the final goal of producing comparative results on the minimum time trajectory taken by two cars. But because the existing program had multiple methods, consisting of significant combinations of optimization parameter, my program cannot account for all of them. There are a good number of combination that will prevent the program from successfully executing.
On the topic of complexity, I do concede to the fact that there are



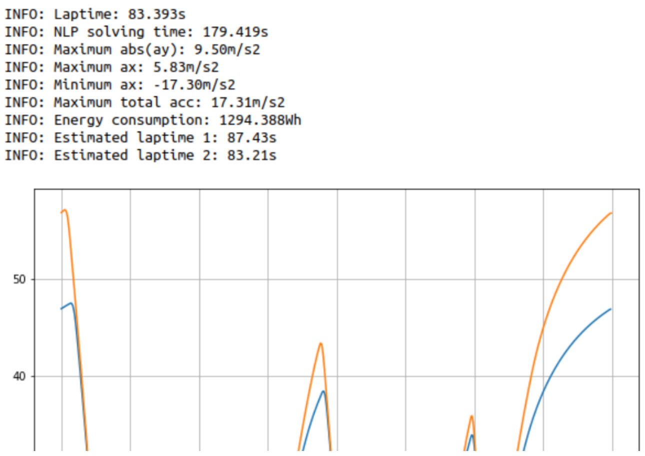
```
INFO: Laptime: 83.393s
INFO: NLP solving time: 177.974s
INFO: Maximum abs(ay): 9.50m/s2
INFO: Maximum ax: 5.83m/s2
INFO: Minimum ax: -17.30m/s2
INFO: Maximum total acc: 17.31m/s2
INFO: Energy consumption: 1294.388Wh
INFO: Estimated laptime 1: 83.17s
INFO: Estimated laptime 2: 83.32s
```

Fig. 6. My program outputting which Car performed better during the 2022 car vs 2019.



```
INFO: Finished export of trajectory: 23:53:23
Car 1 has a better time, and it takes the following trajectory to do so
```

Fig. 7. The raceline taken by 2022 F1 car in Berlin's circuit.



```
INFO: Laptime: 83.393s
INFO: NLP solving time: 179.419s
INFO: Maximum abs(ay): 9.50m/s2
INFO: Maximum ax: 5.83m/s2
INFO: Minimum ax: -17.30m/s2
INFO: Maximum total acc: 17.31m/s2
INFO: Energy consumption: 1294.388Wh
INFO: Estimated laptime 1: 87.43s
INFO: Estimated laptime 2: 83.21s
```

Fig. 8. Car 1 (Honda Civic Type R) vs Car 2 (RB16).

better optimization and simulation algorithm out there, just from the sheer amount of development that goes into building games and game engines. That added with the fact that car manufacturers and racing teams have always tried to get their hands on the best simulator that would account for almost anything in the physical world.

However, since my solution was not to produce a better optimization solution, I did not get to really play with the algorithm so did not have to work my mathematical muscle too much. But I did have to learn new functions and new ways to work with a data structure through this project. I was able to successfully encapsulate another group's work to work in accordance to my convenience.

7 FUTURE IDEAS

One of the first things to do with this program would be to have it optimized for all the possible variations of the simulation. As mentioned before, my comparison is possible for minimum time optimization in this program and not for the other methods and optimization option. So, having that would add to the versatility my program already tries to provide.

Furthermore, to have the inputs be more readable to the user, having a fully graphical interface, possibly with a preview model and a slider to adjust input values would be a lot more helpful.To make this program more accessible and to get more benchmarks for the program, having it hosted as a web application would be a good idea as well. The web application would be highly benefited by an option like that. On the topic of making the entire process of running this program, more readable and user-friendly, some of the output type needs to be represented in forms other than a graph. That could also include a better graph itself, like using the zoom function in MatLabPlot to build an algorithm that checks for the most significant differences and outputs it out.

Finally, depending on the rate I achieve the above mentioned additions, I'd then like to focus on understanding the mathematics behind the optimization call to extend this program to account for aerodynamic output to simulate a car running behind another one. This add another dimension of complexity to the model, but would give a more true to real-life simulation. This would provide justice to the new F1 car that was not really built for this simulator.

ACKNOWLEDGEMENT

All in all, this was a fun project that stemmed from a little curiosity, and I'm glad that I was able to work on it for a college project. I would like to thank Miami's ECE deparment and Prof. Peter Jamieson for formulating this assignment.

REFERENCES

Markus Lienkamp. 2021. Global Race Trajectory Optimization. https://github.com/TUMFTM/global_racetrajectory_optimization
Ying Xiong. 2010. Racing Line Optimization.