Multi-Vehicle Racing Line Optimization

Shang Shi
shangs@andrew.cmu.edu

Srikumar Brundavanam
vbrundav@andrew.cmu.edu

Nikhil CG
nchinnal@andrew.cmu.edu

***Abstract*—A multi-vehicle racing line optimization strategy aimed at controlling two vehicles simultaneously in autonomous racing competitions is presented in this paper. The proposed control strategy is concerned with both cars completing the track in a given time competing for the optimal racing line. The optimization problem considers both the vehicle's dynamics and upper and lower bounds for the states and vehicle controls. Furthermore, the approach allows the cars to safely race on a track while avoiding collision with each other in predefined trajectories. Simulations of both cars following the trajectories are shown using Meshcat. Although the control strategy allowed both cars to follow the given trajectories, there were minimal signs of the two vehicles battling for the optimal trajectory. This needs to be further investigated in future works. The code is released as open-source making it possible to replicate and obtain the current results for further development. The Github Repo can be accessed here.**

***Index Terms*—IPOPT, DIRCOL, Hermite Simpson, Optimal Trajectory, Vehicle Dynamics**

I. Introduction

In recent years, control strategies for autonomous cars have been of significant interest to researchers. Motivated by the desire to have the controls for autonomous vehicles to be optimized for any trajectory, there has been a rise in research in this domain. However the true test for the controller is being optimized to perform in the highest level of control. This led to the rise in research focusing towards designing the controller for Formula 1 racing. Formula 1 is the pinnacle of motor sport. It is the fastest and most technologically advanced racing series. So the motivation of this project was to design a controller for an autonomous version of Formula 1.

In Formula 1 races, each track has an ideal line that all cars tend to follow, known as the racing line [1]. This is the trajectory that race cars follow to maintain the highest speeds at each point of the track. However, since there are multiple cars simultaneously competing for this optimal trajectory, it is not possible to let all the cars follow the same path at the same time. This leads to the race cars battling for position and attempting to pass each other while attempting to follow the optimal trajectory.

This project simulates a scenario with two cars battling for an optimal trajectory. To achieve this while maintaining collision, acceleration, and position constraints, Interior Point Optimizer (IPOPT) with Direct Collocation (Dircol) is used.

To simulate the dynamics of the vehicle, two different methods are used. The kinematic model is the simplest version and the dynamic bicycle model is more realistic.

This paper aims to present the following:

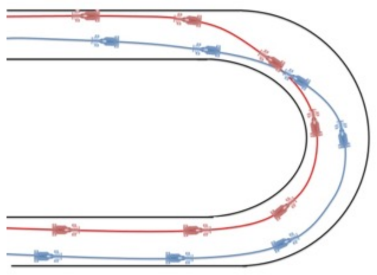
Fig. 1. Racing Line in Formula 1

1) Review the current state of controllers for autonomous cars
2) Present the simplified bicycle model to represent the vehicle dynamics model
3) Introduce constraints to the model like collisions, positional bounds, IPOPT and DIRCOL
4) Simulate the model with both Kinematic and Dynamic approaches

The rest of the paper is organized as follows, Sec II introduces the different methods used in this paper. Sec III talks about the preliminary simulation results, and Sec IV discusses the final simulation results. Subsequently, the visualization, discussion, conclusion and future work are presented in Sec V, VI and VII.

II. Methods

A. IPOPT

IPOPT is a tool for solving non-linear optimization. It is used to solve problems in the below form:

$$\sum_{x\in R^n} f(x)$$

$$st. \quad g^l \leq g(x) \leq g^U$$

$$x^l \leq x \leq x^U$$

x is the optimization variable with lower and upper bounds of x^l and x^U. g(x) are the general nonlinear constraints. Both f(x) and g(x) can be either linear, nonlinear, convex or non convex. This allows for the implementation of the necessary collision, state and control constraints.

B. Dircol

Dircol is a direct non-linear optimization method where the dynamics are enforced as equality constraints between knot points. Most collocation methods uses cubic splines for state trajectories and piece-wise linear interpolation for the controls. In this method, an implicit Hermite Simpson Integrator is used to simulate the dynamics.

C. Hermite Simpson Integration

Hermite-Simpson is an implicit integrator that is used with Dircol. The calculation is as follows:

$$f(x_{k+1/2}), u_{k+1/2}) + \frac{3}{2h}(x_k + x_{k+1})$$
$$-\frac{1}{4}(f(x_k, u_k) + f(x_{k+1}, u_{k+1}) = 0$$

In this method, $f(x_k, u_k)$ and $f(x_{k+1}, u_{k+1})$ can be reused at adjacent steps there is approximately a 50 percent saving on computational cost.

D. Vehicle Dynamic Models

The bicycle model as shown in Figure 2 is used to model the car as the roads are assumed to be flat and the tires maintain good traction with the road.

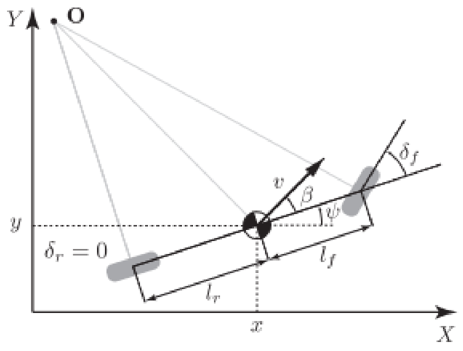

Fig. 2. Bicycle Model

There are two different bicycle models. The kinematic and dynamic models.

1) Kinematic Model: The continuous time equations for a kinematic bicycle model [2] are as below:

$$\dot{x} = v * cos(\psi + \beta)$$
$$\dot{y} = v * sin(\psi + \beta)$$
$$\dot{\psi} = \frac{v}{l_r} sin(\beta)$$
$$\beta = tan^{-1}(\frac{l_r}{l_f + l_r} tan(\delta))$$

The x, y and ψ are the states that describe the vehicle. x and y are the coordinates of the center of mass while ψ is the heading angle of the vehicle. l_f and l_r are the distance from the center of mass to the front and rear axles. In this project, they are assumed to be identical. β is the angle of the current velocity of the center of mass with respect to the longitudinal axis of the vehicle. v and δ are the controls to the system where v is the velocity and δ is the steering angle. This is a very basic model and is computationally very inexpensive. However, the downside to this is that velocity is directly being controlled instead of acceleration which is not realistic for an actual vehicle. The same statement can be made for δ. The steering angle is directly controlled which may lead to unrealistic changes in angle. Therefore, the kinematic model is used only as an initial trial.

2) Dynamics Model: The continuous time equations for a Dynamic bicycle model are as below:

$$\dot{x} = v * cos(\psi + \beta)$$
$$\dot{y} = v * sin(\psi + \beta)$$
$$\dot{\psi} = \frac{v * cos(\beta) * tan(\delta)}{lf + lr}$$
$$a = \dot{v}$$
$$\beta = tan^{-1}(\frac{l_r * \delta}{l_f + l_r})$$

In this model, the controls are acceleration a and steering angle rate $\dot{\delta}$. The states are x position, y position, heading angle ψ, steering angle δ and velocity v. In this model, the controls are more realistic since acceleration can be comparable to how hard someone steps on the gas and the steering angle rate ensures a continuous change in steering angle.

III. Preliminary Simulation

Some basic simulations were ran to experiment with simple passing behaviors with the kinematic and dynamic bicycle models. The scenario set up includes two cars, red and blue as shown in Figure 3. The red car starts ahead of the blue car but at the end, the blue car is required to end up in front. This means that the blue car must pass the red car during the process to avoid any collisions.

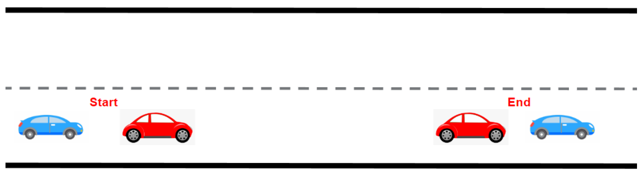

Fig. 3. Simple passing scenario

The red car is given a set trajectory to follow and is constrained to stay on that path. Therefore, its control input is limited to the velocity or acceleration. The steering angle is kept constant. This is done to simulate a normal driving situation where once car changes lane to pass the slower vehicle which drives in a straight line.

The quadratic cost function used for tracking the reference trajectory is as shown below:

$$J = \sum_{i=1}^{N-1} [\frac{1}{2}(x_i - x_{ref,i})^T Q(x_i - x_{ref,i}) + \frac{1}{2} u_i^T R u_i]$$

$$+\frac{1}{2}(x_N - x_{ref,N})^T Q(x_N - x_{ref,N})$$

To enforce collision avoidance, the below constraint is used:

$$norm(pos1 - pos2)^2 \geq R^2$$

pos1 and pos2 represent the x and y positions of car 1 and car 2 respectively. A circle with radius R is used to model the car so the two vehicles do not get within each other's radius. The square is taken on both sides to ensure that it is differentiable everywhere.

1) Kinematic Model: The below Figure 4 demonstrate the passing trajectory with a kinematic model. It can be seen that the blue car starts behind the red car initially but eventually changes lanes and passes the red car.

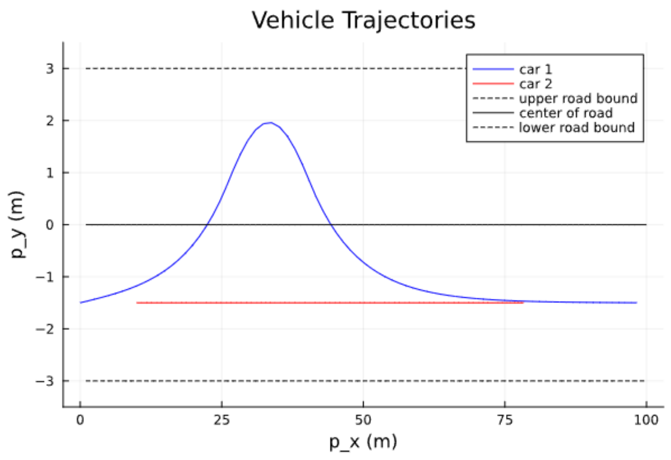

Fig. 4. Kinematic model passing trajectory

From Figure 5, it can be seen that the two cars never collide into each other (y axis = 0) and always maintain a minimum distance between each other. This shows that the collision constraint worked.

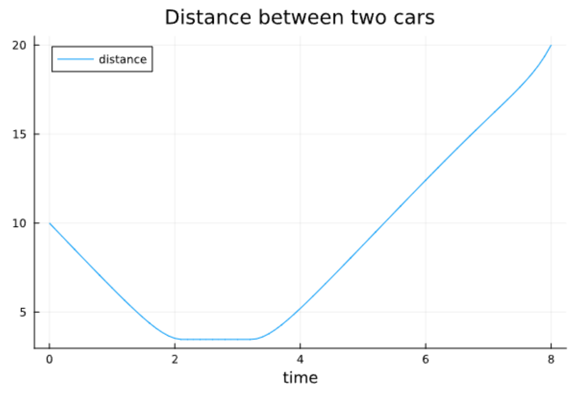

Fig. 5. Distance between vehicles - Kinematic

Figure 6 shows the velocity controls for the system. It can be seen that at the time of passing, the blue vehicle speeds up while the red slows down which is reasonable.

Again, although the results are reasonable, the kinematic model is not ideal as it directly controls the velocity of the vehicle which isn't realistic for actual cars.

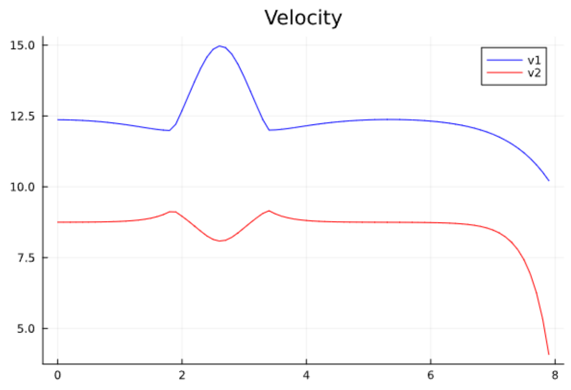

Fig. 6. Velocity controls

2) Dynamic model: The below Figure 7 shows the passing trajectory with the dynamic bicycle model. Not surprisingly, the trajectory is very similar to the kinematic model.

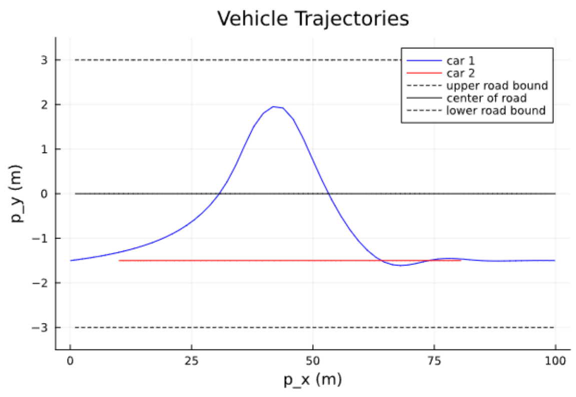

Fig. 7. Dynamic model passing trajectory

In addition, the collision constraint works for this model as well as seen in Figure 8. The vehicles always maintain at least a 2R distance between each other.

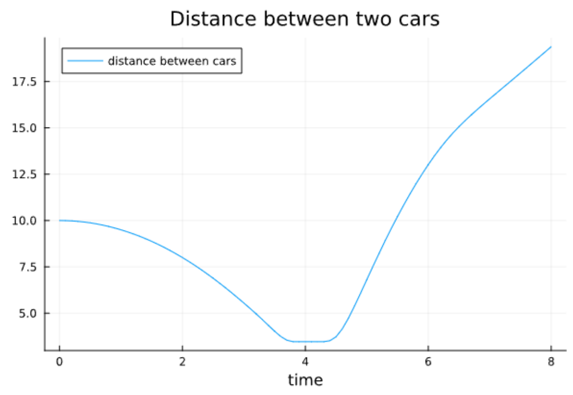

Fig. 8. Distance between vehicles - Dynamic

The main difference from the kinematic model is in the velocity and acceleration.

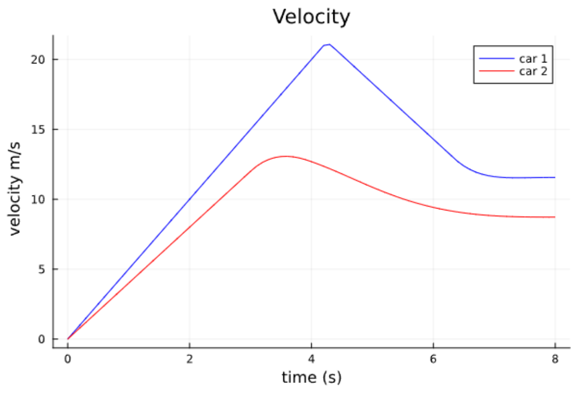

Fig. 9. Velocity of vehicles

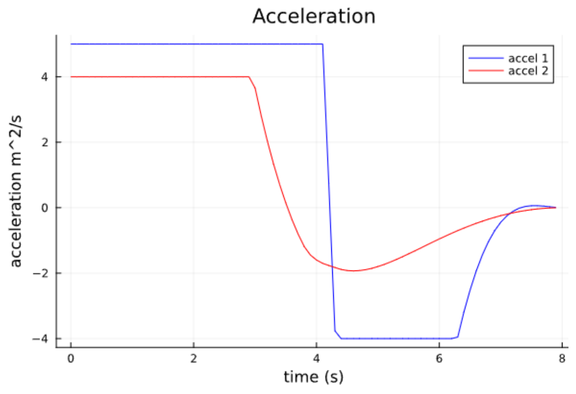

Fig. 10. Acceleration controls

Since acceleration is being directly controlled, a more reasonable velocity plot is obtained. The initial condition is set that both cars were stationary to start with as shown in Figure 9. From there, the blue car accelerates at a faster rate to perform a passing maneuver on the red car. Acceleration and velocity limits are set on the car to mimic a more realistic scenario. However, these are chosen quite arbitrarily. From the acceleration plot, braking behavior can also be seen. This makes sense as once the blue car surpasses the red car, it will slow down to drive at a specific reference velocity to mimic the speed limit. These are all only possible with this dynamic model. Therefore the rest of the simulations are only performed with this model.

3) Double integrator with repulsion: The idea of this simulation is to see whether a simple repulsion term in the closed-loop controls can be used to make a car avoid an obstacle. Here for simplicity, the car dynamics are modeled with a double integrator. The repulsion term d used is as follows:

$$d = \frac{K * (pos_{car} - pos_{obs})}{norm(pos_{car} - pos_{obs})^2}$$

K is a tunable gain. pos_{car} and pos_{obs} and the coordinates of the car and obstacle respectively. the norm term in the denominator will repel the vehicle away from the obstacle in the direction that is the vector from the obstacle to the vehicle. d is then added to the closed-loop controls. This simulation is shown in Figure 11. The reason the repulsion doesn't start until the obstacle is reached in this simulation is that until the car is right next to the obstacle, the repelling term is just trying to slow the vehicle down. This is not a big issue in the context of this project.

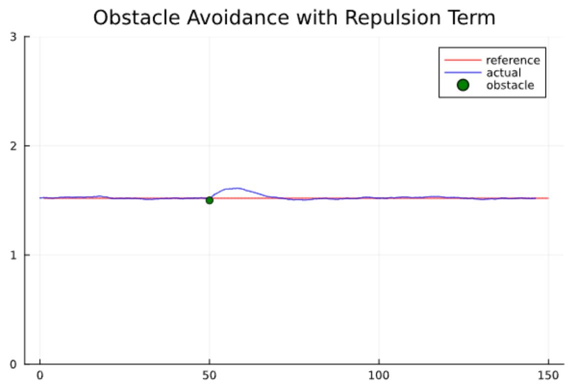

Fig. 11. Repulsion with double integrator

IV. Combined Simulation

From the previous simulations, it can be seen that the effects of two cars battling for the optimal trajectory are not displayed. Therefore to achieve this effect, an ego vehicle with a cost function and collision constraints is used in combination with a second closed-loop controlled vehicle. The effects of this are tested on both a straight and sinusoidal path.

For this simulation, the dynamic bicycle and double integrator models are combined. The blue car (ego vehicle) uses the bicycle dynamic model and the red car (closed loop controls) is simulated with the double integrator model with the repulsion term. The car with double integrator dynamics is controlled using a PD controller. The final states are then stacked with the states of the ego vehicle as shown below. Moreover, as there is a PD controller for controlling the red car, the controls in the combined vehicle dynamics only control the steering angle rate ($\dot{\delta}$) and the acceleration (a) of the blue car (ego vehicle). These combined dynamics are then integrated using Hermite-Simpson and used Dircol and IPOPT to solve for an optimal solution.

$$\dot{x} = \begin{bmatrix} \dot{x}_{ego} \\ \dot{x}_{non-ego} \end{bmatrix}$$

$$u = \begin{bmatrix} a_{ego} \\ \dot{\delta}_{ego} \end{bmatrix}$$

V. Discussion

Figures 12, 13, and, 14 show vehicle trajectories, ego vehicle control usage, and vehicle velocities respectively for a straight path. From Figure 12, it can be observed that the vehicles do not follow the expected straight-line reference trajectory. One of the goals of this simulation is to avoid a collision between the vehicles and this is observed in Figure 12.

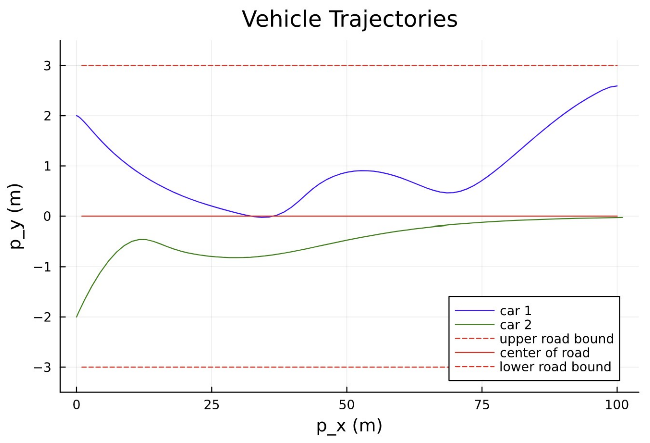

Fig. 12. Vehicle Trajectories for Straight Line

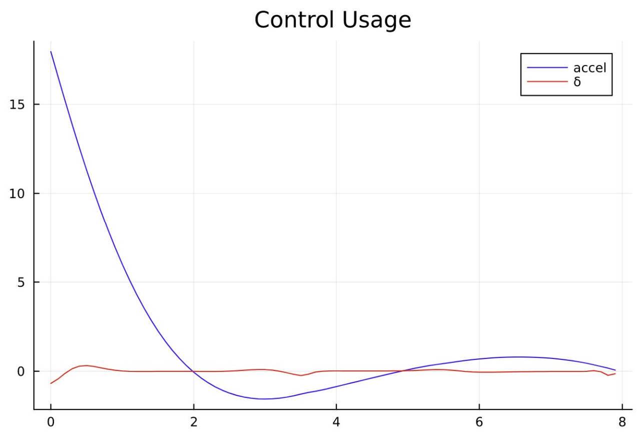

Fig. 13. Control Usage for Straight line

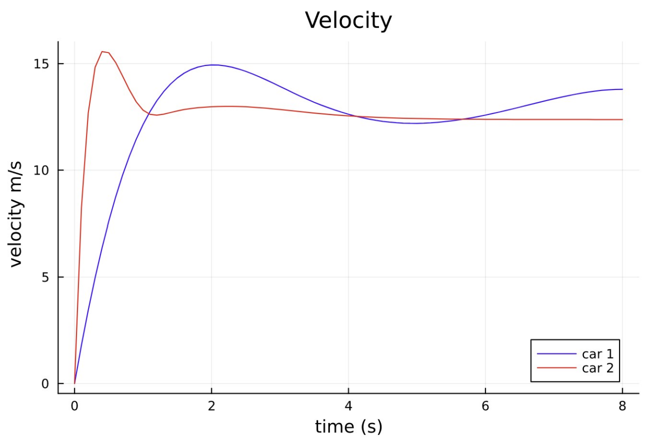

Fig. 14. Vehicle Velocities for a straight path

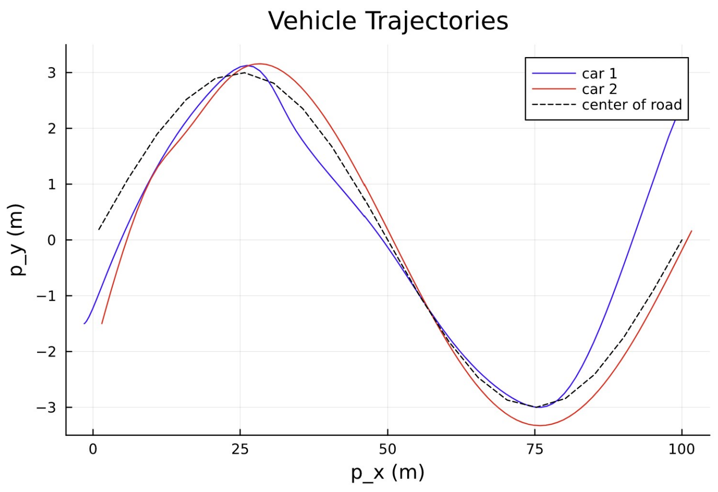

Fig. 15. Vehicle Trajectories for Sinusoidal Path

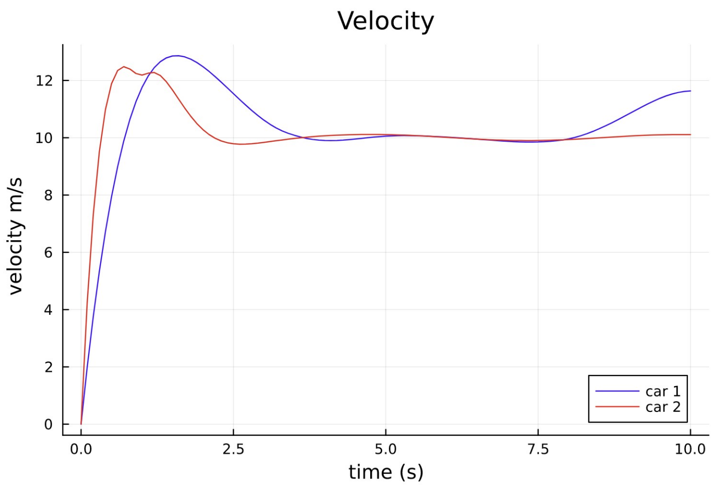

Fig. 16. Vehicle velocity for Sinusoidal Path

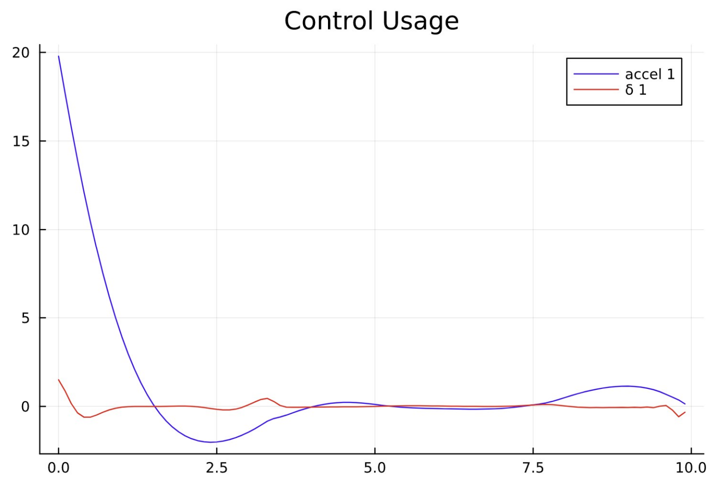

Fig. 17. Control Usage for Sinusoidal Path

From Figure 14, it is shown how the vehicle velocities change over time. It is interesting to note how the velocities reach a maximum as both cars are trying to reach the reference trajectory in the beginning. Both the velocities then stabilize over time indicating little deviation from the reference trajectory. This further goes to show how the battling for the position is not simulated

Figures 15, 17, and 16 show vehicle trajectories, ego vehicle control usage, and vehicle velocities respectively for when following a sinusoidal reference trajectory. Figure 15 shows that while the cars follow the reference trajectories, they do not battle for position as intended. The hypothesis is that both the cars that need to compete for the optimal trajectory fail as one of the cars always stays ahead. This can also be observed from the corresponding vehicle velocities in Fig 17, where one car maintains a higher velocity than the other.

Moreover, in Figure 13 and Figure 17, it is shown that the control parameters stabilize over time and are in agreement with the velocity plot as the velocity eventually stabilizes as well.

VI. Conclusion

In this paper, a Multi-vehicle strategy racing line optimization for autonomous racing scenarios is investigated. Simulations performed using the IPOPT with DIRCOL demonstrated the validity of our method in terms of following trajectories with room for further improvement regarding the simulation of battling for position. The simulations have so far been performed on a straight and sinusoidal track which is not realistic to actual racing tracks. The ego and closed loop controlled vehicles were able to avoid collisions while following the trajectories. However, again, the phenomena of two vehicles battling for position is not displayed.

VII. Future Work

For future work, it is planned to further tune the parameters to investigate the reason why the simulation did not perform as hypothesized. In addition, the second vehicle is being simulated with double integrator dynamics which is not realistic. Therefore a more realistic dynamic model for this vehicle can be implemented. Even for the ego vehicle, more complex modeling of the vehicle dynamics that includes the tire slip angles and traction can be used to form more realistic trajectories. In this project, the reference vehicle trajectories simulated include only a straight line and a sinusoidal trajectory. However, realistic race tracks include far more complex trajectories with unexpected turns. It is planned to further develop our controller to work with such realistic race tracks and observe the feasibility of our method on such paths.

VIII. Acknowledgements

The authors would like to acknowledge the invaluable support we received from Prof. Zachary Machester of Carnegie Mellon University and Graduate Researcher Kevin Tracy at Carnegie Mellon University. We acknowledge that this work would not have been possible without their support, and we are grateful for their contributions to this project.

References

[1] A. Jain and M. Morari. Computing the racing line using bayesian optimization. *CoRR*, abs/2002.04794, 2020.

[2] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, 2015.